Datalogisk Institut
KØBENHAVNS UNIVERSITET

DIKU

Proceedings fra den 15. Danske Konference i Mønstergenkendelse og Billedanalyse

Eds. Søren I. Olsen

Technical Report no. 06/08
ISSN: 0107-8283

Dept. of Computer Science
University of Copenhagen • Universitetsparken 1
DK-2100 Copenhagen • Denmark

Proceedings

fra

Den 15. Danske Konference om

Mønstergenkendelse og Billedanalyse

arrangeret af

Dansk Selskab for Automatisk Genkendelse af Mønstre

Datalogisk Institut, Københavns Universitet

Torsdag d. 24. august og Fredag d. 25 august 2006

Editor Søren I. Olsen

Indholdsfortegnelse:

Multiple View Geometry and Global Optimization

Fredrik Kahl
Centre for Mathematical Sciences (LTH), Lund University

Resumé

In this talk, a framework for solving geometric reconstruction problems in computer vision will be presented based on the L_p-norm of reprojection errors (for various $p >= 1$). That is, the goal is to minimize the differences between reprojected, hypothesized 3D scene geometry and image measurements. While traditional algorithms often suffer from local minima, we pursue the goal of achieving globally optimal solutions. Three different optimization approaches will be considered, namely (i) quasi-convex optimization, (ii) LMI relaxations and (iii) fractional programming. These schemes are applied to a number of classical vision problems including triangulation, camera resectioning and epipolar geometry estimation. The methods have also been validated on real data in different problem settings with small and large dimensions and with good performance.

Automatic Segmentation of the Articular Cartilage in Knee MRI Using a Hierarchical Multi-class Classification Scheme

Jenny Folkesson[1], Erik Dam[1,2], Ole Fogh Olsen[1], Paola Pettersen[2], and Claus Christiansen[2]

[1] Image Analysis Group, IT University of Copenhagen, Denmark
jenny@itu.dk
[2] Center for Clinical and Basic Research, Ballerup, Denmark

Abstract. Osteoarthritis is characterized by the degeneration of the articular cartilage in joints. We have developed a fully automatic method for segmenting the articular cartilage in knee MR scans based on supervised learning. A binary approximate kNN classifier first roughly separates cartilage from background voxels, then a three-class classifier assigns one of three classes to each voxel that is classified as cartilage by the binary classifier. The resulting sensitivity and specificity are 90.0% and 99.8% respectively for the medial cartilage compartments. We show that an accurate automatic cartilage segmentation is achievable using a low-field MR scanner.

1 Introduction

Osteoarthritis (OA) is one of the major health concerns among the elderly today [1]. The main effects of OA is the degradation of the articular cartilage together with remodeling and overgrowth of bone, a process causing loss of mobility of the joints. It typically affects large weight bearing joints as hips and knees. Currently, the treatment of OA is restricted to symptom control, because as yet there are no disease-modifying drugs [2].

MRI allows for quantitative evaluation of the cartilage [3],[4], and cartilage deterioration can be detected using this technique [5]. MRI also has the advantage of being a non-invasive technique.

When assessing the cartilage, the MR scans can be manually segmented slice-by-slice by experts, but for clinical studies manual methods are too time consuming and are also prone to inter- and intra-observer variability. When automating the cartilage segmentation, the main challenges are the thin structure of the cartilage and the low contrast between the cartilage and surrounding soft tissues. The progression of OA is very often slow and it can take many years before the cartilage is reduced from its typical thickness of a few millimeters to possible total loss. It is therefore important to have high accuracy and precision of the cartilage assessment technique in order to detect statistically significant changes. This enables the correlation of the method with the effects of drugs, and the evaluation of their benefit to the joint in reducing the signs of the disease.

Several groups have developed automated methods for cartilage segmentation. 2D methods has limited continuation between slices and since they have to be converted into a 3D segmentation when finding for example thickness maps, it is advantageous to perform segmentation in 3D directly. Among the 3D techniques that have been developed, Grau et al. [6] use a semi-automatic segmentation method that is based on a watershed approach. The method is evaluated on 7 scans from 4 subjects and has an average sensitivity and specificity of 90.0% and 99.9% respectively. Pakin et al. [7] have developed an automatic segmentation method based on region growing followed by two-class clustering. It is evaluated on one scan with resulting sensitivity and specificity of 66.2% and 99.6%. The semi-automatic segmentation method of Warfield et al. [8], [9] iterates between a classification step and a template registration step, and has a lower variability compared to repeated manual segmentations on the scan it was evaluated on. Naish et al. [10] use a data set that consists of a longitudinal study of OA patients and local intensity changes over time is used as a measure of cartilage degradation. However, the cartilage is manually or semi-automatically segmented.

All of the methods mentioned (except for the one of Naish et al. but they have not focused on the segmentation part) have only been evaluated on a handful of scans, and the only fully automatic segmentation produces low sensitivity and specificity values compared to the semi-automatic methods.

In this paper, we present a method for segmenting the tibial and femoral medial cartilage in 3D MR scans of knees. The segmentation is based on an three class approximate kNN classification scheme and is improved by selecting the largest connected component from the result of the classification. The segmentation method works directly in 3D, not in 2D slices, and is fully automatic. This is an improvement of previous work [11] which was a method for locating tibial medial cartilage for the initialization of a shape model, a method based on a two class kNN classifier without any feature selection incorporated.

Our segmentation algorithm aids the automatization of cartilage assessment and is intended for clinical studies on a low-field MR scanner. Though the image quality of the scanner we are using is slightly lower compared to the conventional high-field scanners, we propose to examine if accurate automatic cartilage segmentation is achievable also on a low-field scanner. If such a scanner can be used in clinical studies it would reduce the costs significantly. It has been shown that low-field dedicated extremity MRI can provide similar information on bone erosions and synovitis as expensive high-field MRI units [12] comparing manual segmentations, but there has to our knowledge not been published any work on automatic segmentation of cartilage on low-field MRI. From the automatic segmentation, relevant quantitative measures such as the cartilage volume and thickness can be calculated either globally or locally in a point or a small area. In the latter case comparison between patients or temporal studies of the same patient will require establishing geometric or anatomical correspondence either by expert annotations or by automated modeling of landmarks. An automated approach for this is planned for future work and will not be part of this paper.

2 Methods

2.1 Image Acquisition

An Esaote C-Span low-field 0.18 T scanner dedicated to imaging of extremities acquires Turbo 3D T1 scans (40° flip angle, T_R 50 ms, T_E 16 ms). The scans are made through the sagittal plane with a voxel size in the range $0.7031 \times 0.7031 \times (0.7031/0.7813/0.8594)\, mm^3$. Among the total of 71 scans, 50 have the resolution $0.7031 \times 0.7031 \times 0.7813\, mm^3$, 3 the resolution $0.7031 \times 0.7031 \times 0.7031\, mm^3$ and the remaining 18 scans have the resolution $0.7031 \times 0.7031 \times 0.8594\, mm^3$. The scans all have the size $256 \times 256 \times 104$ voxels, but we only use the central $170 \times 170 \times 104$ voxels because only they contain information.

The scans have been manually segmented on a slice-by-slice basis by a radiologist. A scan slice with the tibial and femoral medial cartilage manually segmented is shown in Figure 1.

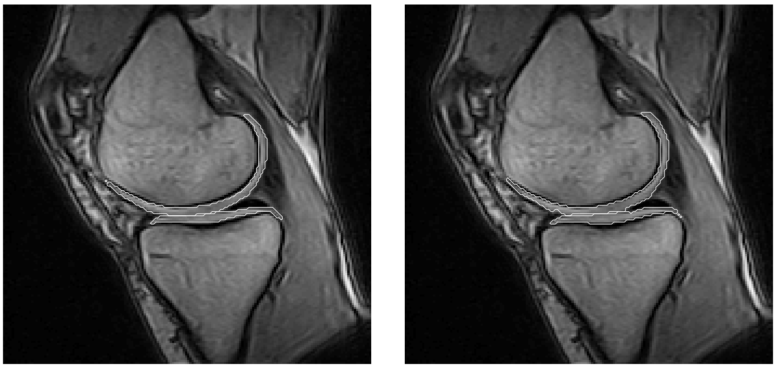


Fig. 1. To the left, a slice from a knee MR scan where the tibial medial and femoral medial cartilage is segmented manually by radiologists. The size of this slice is 170x170 pixels. To the right is the result from our automatic segmentation for the corresponding slice. The sensitivity and specificity for this scan are 92.52% and 99.82% respectively, with a dice similarity coefficient of 0.83.

The 71 scans in the data set are of both left and right knees. In order to treat all scans analogously, the right knees are reflected about the center in the sagittal plane. The test subjects are both males and females aged between 21 and 72 years. They have no or mild OA symptoms, diagnosed by radiologists as being between 0 and 3 on the Kellgren and Lawrence Index [13].

2.2 Cartilage Classification

For the segmentation of cartilage we use an approximate *k*NN classifier, which is implemented in an Approximate Nearest Neighbor (ANN) framework developed by Mount and colleagues [14]. The ANN classifier is in principle the same as a

kNN classifier, but with the modification that you can allow for a small amount of error in the search for nearest neighbors which may improve the run time significantly. An error bound, ϵ, is introduced, so instead of returning the k nearest neighbors from a data set, the ANN search algorithm returns k points such that the ratio of the distance between the ith reported point ($1 \leq i \leq k$) and the true ith nearest neighbor is at most $1 + \epsilon$. We have found empirically that examining the 100 nearest neighbors yields a good balance between computational complexity and accuracy, and we set $\epsilon = 2$, a value that only marginally lowers the accuracy while reducing computational time significantly.

In this work we examine the medial cartilage since OA is more often observed in this compartment [15] and in particular in the medial tibial part [16], thus these compartments are of major interest when it comes to finding disease markers for OA. In order to separate different types of cartilage from one another we use a three class classifier, where the classes are tibial medial cartilage, femoral medial cartilage and background.

The classification is hierarchical, and the first step is a two class classification where the voxels are roughly separated into cartilage or background. The kNN produces class probabilities for every voxel, and in this step we set the threshold at 0.65 yielding a sensitivity for medial cartilage close to 99%. This also results in a large amount of false positives, but since typically only a few percent of the total volume within the knee belongs to the cartilage, this first step is a way of reducing data significantly. In the second step, the voxels classified as cartilage in the first step are reconsidered. This time we use a three class classifier, where the three classes are tibial and femoral medial cartilage and background, and class membership is decided based on a majority vote. The three class classifier contains more features and the features are optimized to separate the three classes whereas the classifier in the first step has features optimized to separate cartilage from background. A sketch of the hierarchical classification scheme is illustrated in Figure 2.

We have also tested a direct partitioning into the three classes, but the hierarchical approach yields better results and is faster, since the first step has less features and thus lower computational complexity. The classifier in the first step has a set of 28 features compared to the three class classifier in the second step that contains 52 features.

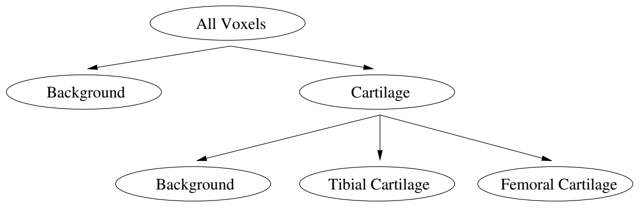


Fig. 2. Tree representation of the hierarchical classification scheme

2.3 Features and Feature Selection

In order to find a feature set that performs well for our classification scheme, we here introduce our set of candidate features and the subsets of the features that were found from our feature selection method [17], which consists of sequential forward selection followed by sequential backward selection.

When a radiologist examines an MR scan for cartilage, she or he takes the location and the intensity in the image into consideration. We therefore consider these as candidate features. Both the raw intensity and the Gaussian smoothed intensities on three different scales $(0.65mm, 1.1mm, 2.5mm)$ are examined.

One can also consider features that are related to the geometry of the object in question. The 3-jet, which is all first, second and third order derivatives with respect to (x, y, z) forms a basis which can describe all geometric features up to third order [18] and are listed as candidate features. All the derivatives mentioned in this section are Gaussian derivatives and are defined as $I_{i_1,\ldots,i_n} = \int \tilde{I}(\bar{x}) D_{i_1,\ldots,i_n} g(\bar{x}, \sigma_1) d\bar{x}$, where g is a Gaussian, D a differential operator and σ_1 is the scale. All features are examined on the three scales, selected to cover the range of different cartilage thicknesses, mentioned above.

Cartilage can be described as a thin curved disc in 3D. The Hessian (H), which is the symmetric matrix containing second order derivatives with respect to the coordinates (x, y, z),

$$H = \begin{pmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{pmatrix},$$

is therefore considered. The eigenvectors of the Hessian points in the directions of the principal curvatures and its eigenvalues corresponds to the curvature in those directions. A thin disc such as cartilage will locally yield one large and two small eigenvalues. The eigenvalues as well as the three eigenvectors are candidate features.

A feature that has been shown to be significant in the detection of thin structures such as fingerprints is the structure tensor (ST) [19]. It is a symmetric matrix containing products of the first order derivatives convolved with a Gaussian,

$$ST = G_{\sigma_2} * \begin{pmatrix} I_x I_x & I_x I_y & I_x I_z \\ I_y I_x & I_y I_y & I_y I_z \\ I_z I_x & I_z I_y & I_z I_z \end{pmatrix},$$

where σ is not necessarily the same scale as the one used for obtaining the derivatives. The ST examines the local gradient distribution at each location (x, y, z). The directions of the eigenvectors depend on the variation in the neighborhood. The eigenvalues and eigenvectors of the ST were considered as potential features with a combination of three scales of σ_1 and three scales of σ_2.

The third order derivatives with respect to (x, y, z) can be conveniently represented in the third order tensor I_{ijk}. Examining the third order structure in the local gradient direction (I_x, I_y, I_z) can be described using Einstein summation as

$$L_{www} = I_{ijk} I_i I_j I_k / (I_i I_i)^{3/2}.$$

The third order tensor is examined in the gradient direction on three different scales, which were considered as possible features.

2.4 Selected Features

The features used in the two class classifier are the position in the image, the Gaussian smoothed intensities on three different scales ($0.65mm, 1.1mm, 2.5mm$) and the raw intensities, the first order Gaussian derivatives on scales $0.65mm$ and $2.5mm$, the eigenvalues and the eigenvector corresponding to the largest eigenvalue of the structure tensor with $\sigma_1 = 0.65mm$ and $\sigma_2 = 2.5mm$, and the eigenvalues of the Hessian on scales $1.1mm$ and $2.5mm$.

The features in the three class classifier consist of combinations of first, second and third order Gaussian derivatives on the three different scales mentioned, the Gaussian smoothed intensities on three different scales ($0.65mm, 1.1mm, 2.5mm$) and the raw intensities, the position, the eigenvector corresponding to the largest eigenvalue of the ST with $\sigma_1 = 0.65mm$ and $\sigma_2 = 1.1mm$, the eigenvalues of the ST with $\sigma_1 = 1.1mm$ and $\sigma_2 = 2.5mm$, the eigenvalues of the Hessian on scales $1.1mm$ and $2.5mm$.

The features selected as most significant are the Hessian and the structure tensor along with the intensity and the position in the image. The features were normalized between zero and one. Normalization for unit variance was also examined, but the normalization of values between zero and one produces slightly better results.

3 Results

From our data set of 71 scans we use 25 for training and 46 for the evaluation of our algorithm. The results of our automatic segmentation is compared to the manual segmentation made by radiologists, resulting in an average sensitivity and specificity of 90.0% (±2.6% st.d.) and 99.8% (±0.06% st.d.) respectively

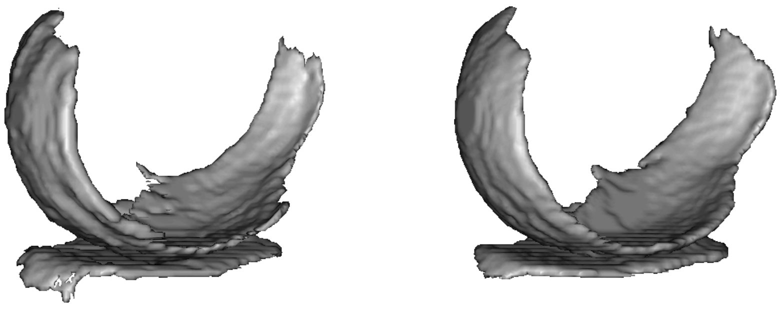


Fig. 3. On the left is the manually segmented medial cartilage from a knee MR scan. To the right is the corresponding automatic segmentation. For this scan, the sensitivity and specificity are 94.82% and 99.79% respectively, with a dice of 0.81.

for the test set for the medial cartilage compartments. A comparison between a golden standard segmentation and an automatically segmented knee MR scan can be seen in Figure 3. A slice by slice comparison is displayed in Figure 1. The dice similarity coefficient (DSC) measures spatial volume overlap between two segmentations, A and B, and is defined as $DSC(A,B) = \frac{2\times|A\cap B|}{|A|+|B|}$. The Dice similarity coefficient between our automatic segmentation and the golden standard segmentation is for the test set on average 0.80 ($\pm$0.03 st.d.).

4 Discussion

The average sensitivity and specificity of our method compared to the results of methods with similar evaluation though on different data is presented in Table 1. Comparing our method with the fully automatic segmentation algorithm (Pakin et al. [7]), we get a distinctly higher sensitivity and a slightly better specificity. Though slightly worse, the sensitivity and specificity of our method are comparable to those of Grau et al. [6]. They have a higher volume overlap (DSC=0.90) however their method is semi-automatic. We have evaluated our segmentation algorithm on more scans than the other two methods. Some of the semi-automated

Table 1

	Our method	Method of Pakin [7]	Method of Grau [6]
Sensitivity	90.01%	66.22%	90.03%
Specificity	99.80%	99.56%	99.87%
Data set	25+46	1	7
Interaction time	0	0	5-10 min

segmentation techniques described in section 1 have been evaluated in terms of inter- and intra-observer variability of the method compared to manual segmentation. As for our method, future work will involve inter-scan variability, and we will also examine intra-user variability for the manual segmentations.

Our segmentation algorithm performs well compared to two leading cartilage segmentation schemes, which leads us to the conclusion that accurate, fully automatic cartilage segmentation is achievable in low-field MR scanners.

References

1. J.Wyngaarden, H.Smith, L., Bennett, J.: Cecil Textbook of Medicine. 19 edn. Volume 2. W. B. Saunders (1992)
2. Creamer, P., Hochberg, M.C.: Osteoarthritis. Lancet **350** (1997) 503–509
3. Muensterer, O., Eckstein, F., Hahn, D., Putz, R.: Computer-aided three dimensional assessment of knee-joint cartilage with magnetic resonance imaging. Clinical Biomechanics **11** (1996) 260–266
4. Graichen, H., Eisenhart-Rothe, R.V., Vogl, T., Englmeier, K.H., Eckstein, F.: Quantitative assessment of cartilage status in osteoarthritis by quantitative magnetic resonance imaging. Arthritis and Rheumatism **50** (2004) 811–816

5. Loeuille, D., Olivier, P., Mainard, D., Gillet, P., Netter, P., Blum, A.: Magnetic resonance imaging of normal and osteoarthritic cartilage. Arthritis and Rheumatism **41** (1998) 963–975
6. Grau, V., Mewes, A., Alcaiz, M., Kikinis, R., Warfield, S.: Improved watershed transform for medical image segmentation using prior information. IEEE Transactions on Medical Imaging **23** (2004)
7. Pakin, S.K., Tamez-Pena, J.G., Totterman, S., J.Parker, K.: Segmentation, surface extraction and thickness computation of articular cartilage. Volume 4684., SPIE (2002) 155–166
8. Warfield, S.K., Kaus, M., Jolesz, F.A., Kikinis, R.: Adaptive, template moderated, spatially varying statistical classification. Medical Image Analysis (2000) 43–55
9. Warfield, S.K., Winalski, C., Jolesz, F.A., Kikinis, R.: Automatic segmentation of mri of the knee. SPL Technical Report 91, ISMRM Sixth Scientific Meeting and Exhibition, Sydney, Australia (1998)
10. Naish, J.H., Vincent, G., Bowes, M., Kothari, M., White, D., Waterton, J.C., J.Taylor, C.: A method to monitor local changes in mr signal intensity in articular cartilage: A potential marker for cartilage degeneration in osteoarthritis. Volume 3217., MICCAI (2004) 959–966
11. Folkesson, J., Dam, E., Pettersen, P., Olsen, O.F., Nielsen, M., Christiansen, C.: Locating articular cartilage in mr images. Volume 5747., SPIE (2005) 1484–1490
12. Ejbjerg, B., Narvestad, E., adn H.S. Thomsen, S.J., Ostergaard, M.: Optimised, low cost, low field dedicated extremity mri is highly specific and sensitive for synovitis and bone erosions in rheumatoid arthritis wrist and finger joints: a comparison with conventional high-field mri and radiography. Annals of the Rheumatic Diseases **13** (2005)
13. Kellgren, J., Lawrence, J.: Radiological assessment of osteo-arthrosis. Annals of the Rheumatic Diseases **16** (1957)
14. Arya, S., Mount, D., Netanyahu, N., Silverman, R., Wu, A.: An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. Number 5, ACM-SIAM. Discrete Algorithms (1994) 573–582
15. Dunn, T., Lu, Y., Jin, H., Ries, M., Majumdar, S.: T2 relaxation time of cartilage at mr imaging: comparison with severity of knee osteoarthritis. Radiology **232** (2004) 592–598
16. Kamibayashi, L., Wyss, U., Cooke, T., Zee, B.: Changes in mean trabecular orientation in the medial condyle of the proximal tibia in osteoarthritis. Calcif Tissue Int. **57** (1995) 69–73
17. Folkesson, J., Dam, E., Olsen, O.F., Pettersen, P., Christiansen, C.: A supervised learning approach to segmenting articular tibial cartilage in knee mri. Technical report, IT University of Copenhagen, Rued Langgaards Vej 7, 2300 Copenhagen S, Denmark (2005)
18. Florack, L.: The Syntactical Structure of Scalar Images. PhD thesis, University of Utrecht (1993)
19. Weickert, J.: Anisotropic Diffusion in Image Processing. B. G. Teubner (1998)

Data mining a functional neuroimaging database for functional segregation in brain regions

Finn Årup Nielsen[*†‡], Daniela Balslev[§], Lars Kai Hansen[†*]

July 3, 2006

Abstract

We describe a specialized neuroinformatic data mining technique in connection with a meta-analytic functional neuroimaging database: We mine for functional segregation within brain regions by identifying journal articles that report brain activations within the regions and clustering the abstract of the articles using non-negative matrix factorization on the bag-of-words matrix. We divide the brain activations reported in the articles according to the cluster assignment and test for difference between the spatial distribution of the sets of activations. Among our findings is that the memory and pain functions are spatially segregated within the cingulate gyrus.

1 Introduction

Meta-analytic-oriented databases in functional neuroimaging, such as the BrainMap [1] and Brede [2] databases, allow for automated data mining [3, 4, 5, 6]. These databases record so-called Talairach coordinates ("locations") [7] from published human brain mapping studies made with, e.g., positron emission tomography and functional magnetic resonance imaging. The locations represent focal brain activations and are each represented by a 3-dimensional coordinate referenced with respect to a "Talairach" brain atlas [7]. Typically a neuroanatomical term is also associated with the location. Apart from the locations the databases contain description of the experiments in the article that can be correlated to the spatial location information: For the Brede database we have used the words from the abstracts [5] and the linkage to a taxonomy of brain functions [6] as the basis for automated meta-analysis.

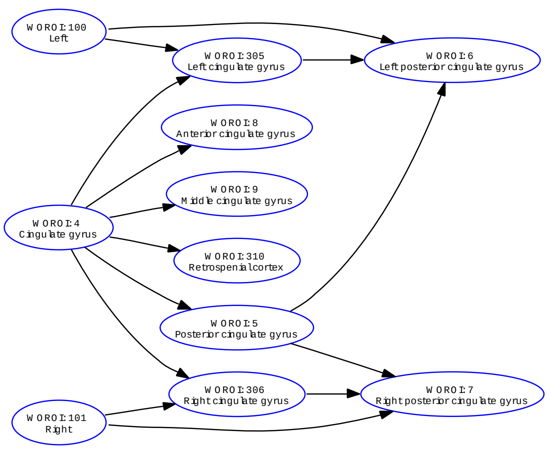


Figure 1: Part of the brain region taxonomy around "cingulate gyrus".

Besides the information from published human brain mapping studies the Brede database has a taxonomy for brain regions, see Fig. 1 for a part of it. It records, e.g., that the "cingulate gyrus" is a subregion of the "cerebral cortex" and that it is a super-region of the "left posterior cingulate gyrus". This hierarchy is partially built from information in the NeuroNames database [8] and the Mai Atlas [9]. It also maintains the variations in the naming, for, e.g., cingulate gyrus they are "cingulate gyri", "gyrus cinguli", "gyrus cingularis" and "cingulate cortex". The taxonomy does probably not capture all relevant variations for many brain regions.

We have previously made a focused data mining on the posterior cingulate brain region using textual data from the PubMed database [10]. In that work we

[*]Lundbeck Foundation Center for Integrated Molecular Brain Imaging

[†]Informatics and Mathematical Modelling, Technical University of Denmark

[‡]Neurobiology Research Unit, Copenhagen University Hospital Rigshospitalet

[§]Danish Research Centre for Magnetic Resonance, Copenhagen University Hospital Hvidovre found that memory and pain (processing) were two prominent functions for posterior cingulate, and that their locations were not equally distributed within this area. Below we will make a similar data mining restricting the analysis to data taken from the Brede database, but expanding the data mining to incorporate the many areas defined in the brain region taxonomy.

2 Method

Our method involves a number of steps that each relies on specific information in our database as well as statistical modeling of relations between the items:

1. Robust kernel density modeling in 3-dimensional brain space for identification of Talairach locations of interest
   (a) Select a brain region.
   (b) Get naming variations for the brain region and all its subregions.
   (c) Get locations that matches one or more of the names.
   (d) Model the distribution with kernel density modeling and discard outliers.
   (e) Include locations that did not match any name but lies in the region.
2. Text mining of abstracts
   (a) Get all abstracts that are associated with the locations.
   (b) Construct a bag-of-words matrix from words in the abstract excluding non-important words.
   (c) Cluster the abstracts
3. Robust multivariate test between sets of Talairach locations.
   (a) Extract locations based on cluster assignment.
   (b) Compare the distribution of set of locations.

We use the data from the Brede database [2] which recorded information from 166 journal articles with a total of 3389 locations. The taxonomy of brain regions contained 313 items. Some of these regions are functional and cytoarchitectonic defined areas and these were ignored. For the rest of the areas steps 1–3 listed above are independently carried out. A final fourth step involves the sorting and intertwining of results from all the brain regions.

After selection of a brain region r (step 1a) we obtain the variations of names from the brain region taxonomy (step 1b). This includes variation of names for the brain region itself as well as all its subregions. For, e.g., "cingulate gyrus" this amounted to 48 different names. We query the database for locations where the neuroanatomical name matches any of the variations (step 1c), and obtain a set of L_r 3-dimensional coordinates that can be represented in a matrix $\mathbf{L}(L_r \times 3)$. We model this data with a kernel density estimator and excluded the 5% most extreme locations in terms of probability density to get rid of outliers (step 1d) [3] giving a smaller set of coordinates. We can augment this smaller set by including coordinates associated with high probability density (step 1e), adding the extra locations that did not match any of the variations in the neuroanatomical names. Some initial tests were made with the inclusion of these locations. Often this would lead to inclusion of location with the label of the neighboring region. There are variation in the application of the Talairach atlas: The locations in the so-called MNI-space are converted by a Brett's piecewise affine transformation [11]. This will exclude some of the variation. However, there still is some overlap between regions when locations are collected across studies. The result presented below are did not include this step.

When we have identified all relevant locations for the brain region r we obtain the abstract of all the articles that contain the locations (step 2a). A bag-of-words matrix $\mathbf{X}(N_r \times P)$ is constructed by counting the frequency of each of the P words in the N_r abstracts (step 2b). A very large stop list is used to exclude ordinary stop words as well as words for neuroanatomy and frequent words not associated with human brain function. To avoid that abstracts where some words occur with high frequency will dominate, the element-wise square root $\sqrt{x_{np}}$ is used in the further processing.

For "clustering" the abstract (step 2c) we perform non-negative matrix factorization (NMF) [12] with an algorithm for updating an "Euclidean" cost function [13]. We factorize the bag-of-words matrix into three matrices plus a residual matrix $\mathbf{U}$ whose Frobenius norm is to be minimized

$$\mathbf{WSH} + \mathbf{U} = \mathbf{X}, \tag{1}$$

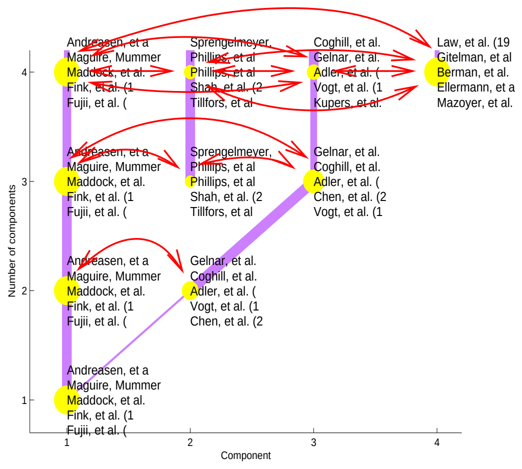


Figure 2: Illustration showing with arrows all the possible comparisons performed between locations in the NMF components (clustered articles). y-axis is size of the NMF K and x-axis the kth component.

where $\mathbf{W}(N_r \times K_r)$ and $\mathbf{H}(K_r \times P_r)$ are non-negative matrices $\mathbf{W} \geq 0$, $\mathbf{H} \geq 0$ and normalized [14], e.g., with the vectorial 2-norm $\|\mathbf{w}_k\|_2 = \|\mathbf{h}_k\| = 1$. $\mathbf{S}(K_r \times K_r)$ is a non-negative diagonal matrix. K_r is the size of the subspace, i.e., the number of components/topics. We distribute the scaling contained in $\mathbf{S}$ equally over the two matrices $\mathbf{W}$ and $\mathbf{H}$

$$\tilde{\mathbf{w}}_k = \mathbf{w}_k \sqrt{s_k} \tag{2}$$

$$\tilde{\mathbf{h}}_k = \mathbf{h}_k \sqrt{s_k}. \tag{3}$$

A winner-take-all function is invoked for exclusive assignment of each abstract n to a component k

$$\breve{\mathbf{w}}_{nk} = \begin{cases} \tilde{w}_{nk} & \text{if } \forall k' \neq k : \tilde{w}_{nk} \geq \tilde{w}_{nk'} \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

A row vector $\mathbf{h}_k$ in the $\mathbf{H}$ matrix contains loading for words on the kth component. The words associated with the highest load are used to label the component.

We vary K_r between 2 and $\tilde{K}_r = \left\lceil \sqrt{\min(N_r, P_r)} \right\rceil$ and thus generate a set of factorized matrices for each brain region r: $\breve{\mathbf{W}}_{K=2,r} \ldots \breve{\mathbf{W}}_{K=\tilde{K},r}$. Each of the matrices contains an assignment of each of the N_r articles to a specific component/topic, and we construct sets of articles for the kth component in the K-sized NMF for the rth brain region:

$$\mathcal{A}_{k,K,r} = \{n : \breve{w}_{n,k,K,r} > 0\} \tag{5}$$

All locations from these articles are extracted. All combination of two sets of location within each brain region and within each of the K-sized NMF are compared, $\mathcal{A}_{k,K,r} \leftrightarrow \mathcal{A}_{k',K,r}$, e.g., for an NMF with five components ($K = 5$) that gives $K!/(2(K-2)!) = 10$ comparisons. We perform this procedure for all the different sizes of NMF subspaces, see Fig. 2 for an illustration with $\tilde{K} = 4$.

For comparison of the distributions between two sets of locations we perform multivariate statistical tests in 3-dimensional Talairach space. We apply the Hotelling's T^2 test [15] and a Monte Carlo permutation test on the "peeling mean" [15, p. 111–112]. The peeling mean provides a robust estimate of the mode by successively deleting the convex hull layers of the data points and taking the mean of the points associated with the last and innermost convex hull [16], see Fig. 3. The permutation test on the peeling mean is performed by randomizing the locations between the two sets. This test is performed since the Hotelling's T^2 is not reliable with non-Gaussian distributed locations.

The Hotelling's T^2 test is applied in two different ways: The first computes the test statistics from the original two sets of locations and the second computes it by first finding the average within each article and then making the test statistics based on the two sets of averages. The latter way is to ensure that an article containing many coordinates in a specific area will not dominate the test statistics. Neither of the three tests allows us to say in which way two sets of coordinates differ.

We use an Internet search engine reporting style, where results are reported in a sorted list with the most relevant information on the top. The results we will present are ordered according to a conjunction P-value, where the resulting P-value is the maximum across the P-values from the three different statistical

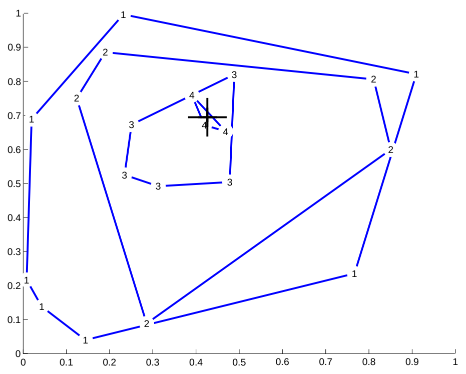


Figure 3: Convex hull peeling mean in 2 dimensions.

```
 #       P-values      (First set) - (Second set) - Brain region
--------------------------------------------------------------------
 1   0.000 0.000 0.000  (pain, painful, 211) - (visual, eye, 565) - Cerebral Cortex (14)
 2   0.000 0.000 0.000  (pain, painful, 230) - (visual, eye, 587) - Telencephalon (13)
 3   0.000 0.000 0.002  (pain, painful, 97) - (memory, retrieval, 141) - Cingulate gyrus (4)
 4   0.000 0.002 0.003  (pain, painful, 269) - (visual, eye, 607) - Forebrain (12)
 5   0.000 0.005 0.000  (expressions, facial, 15) - (recognition, humans, 10) - Amygdala and Hippocampus (202)
 6   0.000 0.004 0.005  (memory, retrieval, 22) - (pain, painful, 5) - Anterior cingulate gyrus (8)
 7   0.000 0.004 0.005  (memory, retrieval, 22) - (pain, painful, 5) - Posterior medial prefrontal cortex (204)
 8   0.000 0.006 0.000  (ear, musical, 5) - (retrieval, faces, 13) - Right frontal lobe (82)
 9   0.000 0.000 0.006  (pain, painful, 100) - (memory, retrieval, 159) - Limbic gyrus (125)
10   0.009 0.002 0.000  (memory, episodic, 27) - (motor, sensorimotor, 20) - Cerebellum (32)
11   0.001 0.004 0.011  (artefacts, categorization, 2) - (memory, word, 28) - Precentral gyrus (68)
12   0.000 0.001 0.015  (pain, painful, 71) - (words, memory, 45) - Limbic lobe (2)
13   0.000 0.000 0.016  (pain, painful, 79) - (memory, episodic, 72) - Prefrontal cortex (22)
14   0.000 0.000 0.024  (artefacts, categorization, 7) - (verbal, visual, 16) - Middle frontal gyrus (148)
15   0.000 0.002 0.029  (memory, episodic, 26) - (pain, painful, 5) - Medial prefrontal cortex (55)
16   0.000 0.031 0.002  (musical, ear, 6) - (artefacts, decision, 10) - Right temporal lobe (86)
17   0.002 0.037 0.009  (pain, noxious, 25) - (motor, visual, 20) - Insula (67)
18   0.000 0.042 0.000  (memory, retrieval, 34) - (pain, painful, 25) - Posterior cingulate gyrus (5)
19   0.006 0.006 0.044  (memory, episodic, 15) - (sensory, visual, 6) - Right fusiform gyrus (134)
20   0.000 0.003 0.047  (visual, emotional, 13) - (faces, familiar, 7) - Left superior temporal gyrus (129)
21   0.000 0.049 0.027  (retrieval, memory, 10) - (rest, memory, 6) - Left anterior cingulate gyrus (94)
22   0.000 0.056 0.006  (memory, episodic, 165) - (artefacts, categorization, 24) - Frontal lobe (18)
23   0.000 0.056 0.042  (facial, faces, 12) - (memory, words, 28) - Left cingulate gyrus (305)
24   0.001 0.039 0.063  (ear, musical, 5) - (artefacts, decision, 10) - Right inferior frontal gyrus (296)
25   0.003 0.070 0.027  (recognition, word, 9) - (eye, attention, 15) - Precuneus (171)
```

Table 1: Automatically generated list of the 25 most relevant functional segregations in brain regions. Columns 2–4 are P-values for the Hotelling's T^2 test with the original coordinates of the locations (column 2) and with the averaged-within-article coordinates used for the test (column 3). Column 4 is the P-value for the peeling permutation test. The words in parentheses are the words associated with the highest load on the components and the number in the parentheses are the number of locations in the component. To the right of the name of the brain region is shown the Brede database identifier for the region.

tests [17].

The data processing uses the Brede toolbox [18], and once the data are entered in the Brede database the entire processing pipeline runs automatically.

3 Results and discussion

Table 1 lists the most relevant functional segregations — one for each brain region. The top entries are in a sense trivial since they indicate a high-level segregation for areas such as "cerebral cortex", "telencephalon" and "forebrain", and the most relevant functional segregation our method reports is between "pain" and "visual". The most frequent words in the abstracts of the Brede database are "visual", "memory", "motor" and "perception", and many of the studies in the Brede database are pain studies. So it is not surprising that we with the Brede database find that major high-level segregation in the brain is between "pain" and "visual". "Pain" locations are mostly distributed in the anterior part of the brain, while "visual" locations have their major share in the posterior part.

Apart from this high-level segregation the most prominent functional segregation appears in the "cingulate gyrus" between pain and memory. Fig. 4 shows the locations for this area colored according to component. A number of subregions and super-region to this area also appear with a segregation between these two functions: "anterior cingulate gyrus", "posterior medial prefrontal cortex", "limbic gyrus" and "posterior cingulate gyrus". Many of the studies that make up these areas were included in connection with our previous study of posterior cingulate [5], where this segregation was identified, and it is thus not surprising that this is refound.

The compound region "amygdala and hippocampus" is segregated into "expressions" and "recognition", and corresponds to a well known functional division in the medial temporal lobe where the hippocampus area is mainly associated with memory whereas the amygdala is involved in the processing of emotional stimuli such as facial expression [20].

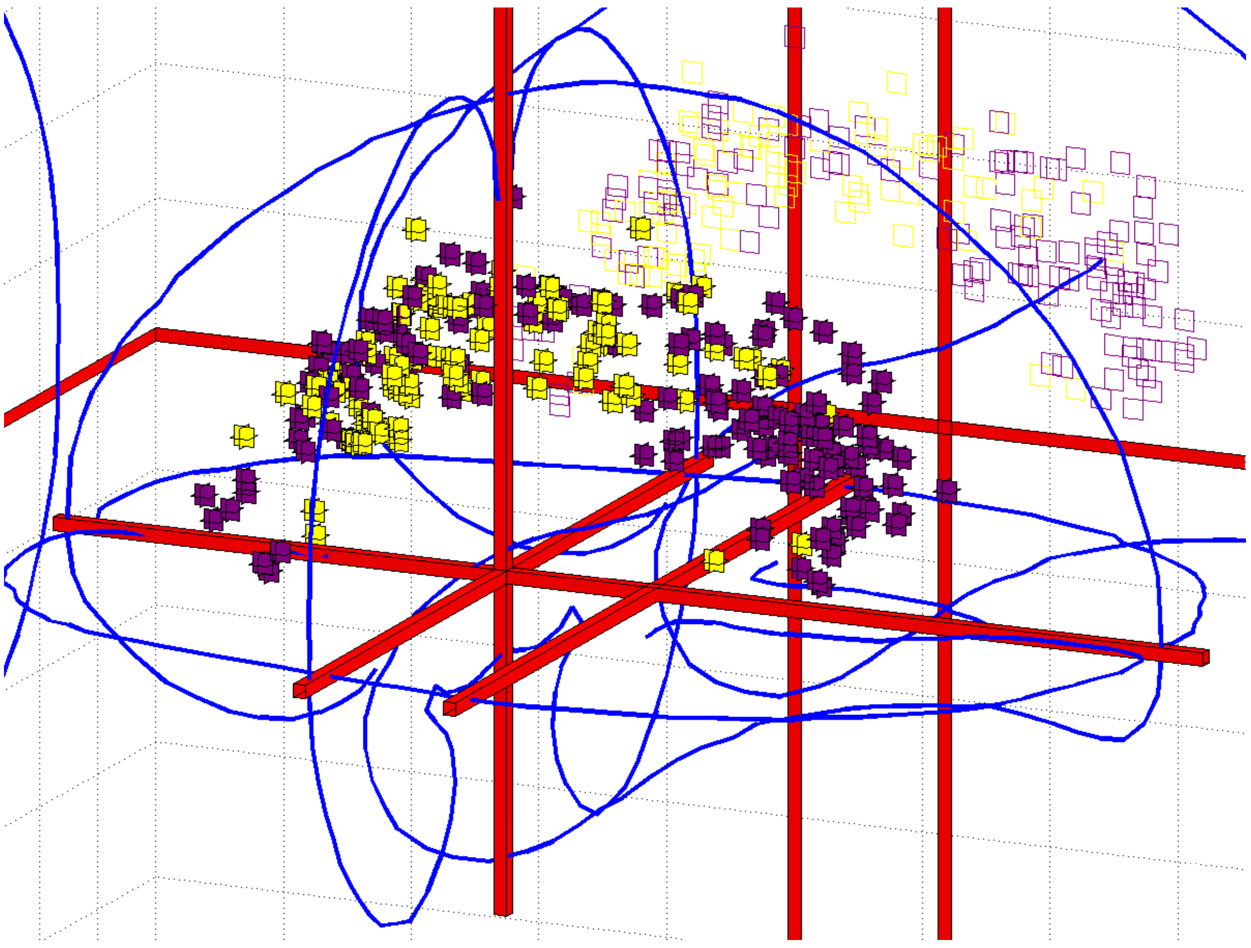


Figure 4: The most relevant functional segregation for "cingulate gyrus" within the Brede database: memory (dark/magenta) and pain (light/yellow). The two sets of locations are plotted in a Corner Cube Environment where each location is represented by a glyph in 3-dimensional space and projected onto "walls" [19] (In this plot only the sagittal projections are visible). The view is from back upper left.

Our method has shortcomings, e.g., some of the results are affected by a number of studies from a single group that investigates artefacts and categorization and reports many activations in specific parts of the brain across articles. Since approximately the same wording is used the abstracts are clustered together, and when the locations from the associated articles are extracted these are spatially clustered often giving rise to segregation when tested against other sets of locations. Furthermore, all the words in a specific article will be modeled together with all locations in that article, e.g., for "cerebellum" a segregation between "memory" and "motor" is found. Actually the "memory" studies have some kind of movement response — overt speech or button pressing — and this is probably why the memory studies activate in cerebellum.

4 Conclusion

We have devised a method that mines a neuroimaging database to extract the main functional modules within a brain region. Such a tool would allow the individual researcher to access the growing base of knowledge generated by the functional imaging studies.

References

[1] Peter T. Fox and Jack L. Lancaster. Neuroscience on the net. *Science*, 266(5187):994–996, November 1994.

[2] Finn Årup Nielsen. The Brede database: a small database for functional neuroimaging. *NeuroImage*, 19(2), June 2003. Presented at the 9th International Conference on Functional Mapping of the Human Brain, June 19–22, 2003, New York, NY. Available on CD-Rom.

[3] Finn Årup Nielsen and Lars Kai Hansen. Modeling of activation data in the BrainMap$^{\text{TM}}$ database: Detection of outliers. *Human Brain Mapping*, 15(3):146–156, March 2002.

[4] Finn Årup Nielsen and Lars Kai Hansen. Finding related functional neuroimaging volumes. *Artificial Intelligence in Medicine*, 30(2):141–151, February 2004.

[5] Finn Årup Nielsen, Lars Kai Hansen, and Daniela Balslev. Mining for associations between text and brain activation in a functional neuroimaging database. *Neuroinformatics*, 2(4):369–380, Winter 2004.

[6] Finn Årup Nielsen. Mass meta-analysis in Talairach space. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 985–992, Cambridge, MA, 2005. MIT Press.

[7] Jean Talairach and Pierre Tournoux. *Co-planar Stereotaxic Atlas of the Human Brain*. Thieme Medical Publisher Inc, New York, January 1988.

[8] Douglas M. Bowden and Richard F. Martin. NeuroNames brain hierarchy. *NeuroImage*, 2(1):63–84, 1995.

[9] Jürgen K. Mai, Joseph Assheuer, and George Paxinos. *Atlas of the Human Brain*. Academic Press, San Diego, California, 1997.

[10] Finn Årup Nielsen, Daniela Balslev, and Lars Kai Hansen. Mining the posterior cingulate: Segregation between memory and pain component. *NeuroImage*, 27(3):520–532, 2005.

[11] Matthew Brett. The MNI brain and the Talairach atlas. http://www.mrc-cbu.cam.ac.uk-/Imaging/Common/mnispace.shtml, February 2002. Accessed 2005 April 9.

[12] Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, October 1999.

[13] Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, *Advances in Neural Information Processing Systems 13: Proceedings of the 2000 Conference*, pages 556–562, Cambridge, Massachusetts, 2001. MIT Press.

[14] Wie Xu, Xin Liu, and Yihong Gong. Document clustering based on non-negative matrix factorization. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 267–273, New York, NY, USA, 2003. ACM Press.

[15] Kantilal Vardichand Mardia, John T. Kent, and John M. Bibby. *Multivariate Analysis*. Probability and Mathematical Statistics. Academic Press, London, 1979.

[16] C. Bradford Barber, David P. Dobkin, and Hannu T. Huhdanpaa. The Quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, 22(4):469–483, December 1996.

[17] Matthrew Brett, Tom Nichols, Jesper Andersson, Tor Wagner, and Jean-Baptiste Poline. When is a conjunction not a conjunction? *NeuroImage*, 22, 2004. Presented at the 10th Annual Meeting of the Organization for Human Brain Mapping, June 14–17, 2004, Budapest, Hungary. Available on CD-ROM.

[18] Finn Årup Nielsen and Lars Kai Hansen. Experiences with Matlab and VRML in functional neuroimaging visualizations. In Scott Klasky and Steve Thorpe, editors, *VDE2000 - Visualization Development Environments, Workshop Proceedings, Princeton, New Jersey, USA, April 27–28, 2000*, pages 76–81, Princeton, New Jersey, April 2000. Princeton Plasma Physics Laboratory.

[19] Kelly Rehm, Kamakshi Lakshminarayan, Sally A. Frutiger, Kirt A. Schaper, De Witt L. Sumners, Stephen C. Strother, Jon R. Anderson, and David A. Rottenberg. A symbolic environment for visualizing activated foci in

functional neuroimaging datasets. *Medical Image Analysis*, 2(3):215–226, September 1998.

[20] Ian Q. Whishaw and Bryan Kolb. *Fundamentals of human neuropsychology.* Worth Publishers, 4th edition, July 1995.

Detection of Connective Tissue Disorders from 3D Aortic MR Images Using Independent Component Analysis

Michael Sass Hansen[1,4], Fei Zhao[1], Honghai Zhang[1], Nicholas E. Walker[2], Andreas Wahle[1], Thomas Scholz[3], and Milan Sonka[1]

[1] Department of Electrical Engineering, University of Iowa, Iowa City, IA 52242, USA
[2] Department of Internal Medicine, University of Iowa, Iowa City, IA 52242, USA
[3] Department of Pediatrics, University of Iowa, Iowa City, IA 52242, USA
[4] Department of Informatics and Mathematical Modelling, Technical University of Denmark, 2800 Lyngby, Denmark

Abstract. A computer-aided diagnosis (CAD) method is reported that allows the objective identification of subjects with connective tissue disorders from 3D aortic MR images using segmentation and independent component analysis (ICA). The first step to extend the model to 4D (3D + time) has also been taken. ICA is an effective tool for connective tissue disease detection in the presence of sparse data using prior knowledge to order the components, and the components can be inspected visually.
3D+time MR image data sets acquired from 31 normal and connective tissue disorder subjects at end-diastole (R-wave peak) and at 45% of the R-R interval were used to evaluate the performance of our method. The automated 3D segmentation result produced accurate aortic surfaces covering the aorta. The CAD method distinguished between normal and connective tissue disorder subjects with a classification accuracy of 93.5 %.

1 INTRODUCTION

Aortic aneurysms and dissections are the 15th leading cause of death in the the U.S., representing 0.7 % of all deaths in 2004 [1]. Persons with certain connective tissue disorders, such as Marfan's syndrome and Familial Thoracic Aortic Aneurysm syndrome are at increased risk of developing aortic aneurysm and dissection, which makes an early detection very important .

This study is approaching cardiovascular disease diagnosis using magnetic resonance (MR) imaging. Producing manual outlining of the aorta in 3D images requires expert knowledge and is a tedious and time-consuming task. Detection of connective tissue disorder is based on a crude diameter measure of the ascending aorta from a single 2D MR-slice. Figure 1 shows three 2D slices of a typical 3D cardiac MR images with manually traced aorta contours.

The aortic segmentation of computed tomography (CT) and MR images has already undergone a lot of research. Rueckert [2] used Geometric Deformable

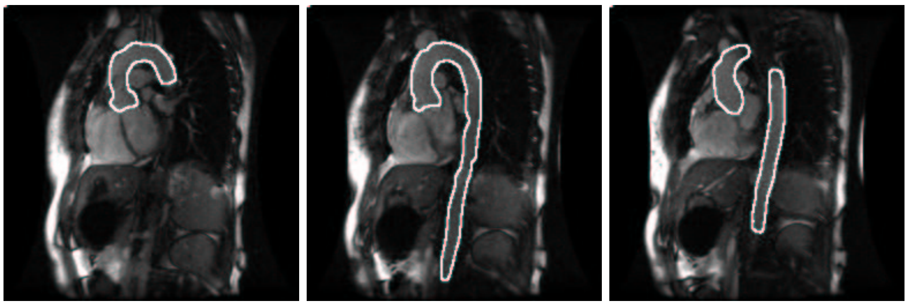


Fig. 1. Three sample 2D slices of a typical aorta candy-cane MR image with manually traced contours outlining aortic lumen.

Models (GDM) to track the ascending and descending aorta. Behrens [3] obtained a coarse segmentation using a Randomized Hough Transform (RHT). Bruijne [4] introduced Adapting Active Shape Models (ASM) for tubular structure segmentation. Subasic [5] utilized the level-set algorithm for segmentation of abdominal aortic aneurysm (AAA). Though aortic segmentation has been repeatedly attempted in the past, we believe this is the first study investigating its use for connective tissue detection. We report a computer-aided diagnosis (CAD) method for objective identification of subjects with connective tissue disorders from 16-phase 3D+time aortic MR images using independent component analysis (ICA).

2 METHODS

Our CAD method consists of three main stages – aortic segmentation, landmarking of the aortic shape and connective tissue disorder diagnosis using ICA. This paper focuses on the ICA-based diagnosis process. The results of the 3D image segmentation were reported previously [6] and are provided here for completeness. The surface segmentation of the aortic lumen is obtained with an automatic 3D segmentation method described in Sect. 2.1. ICA is performed on the aortic 3D shape to provide better descriptors that are visually inspectable, for use in the disease classification step.

2.1 Segmentation

The 3D segmentation algorithm consists of the following three stages:

1. *Aortic surface presegmentation.* A fast marching level set method yields an approximate spatial segmentation of the aorta.
2. *Centerline extraction.* Aortic centerline is obtained by skeletonization.
3. *Accurate aortic surface segmentation.* Accurate aorta surface results from the application of a 2D optimal border detection method.

Aortic surface presegmentation A 3D fast marching segmentation method was used to obtain an approximate aortic surface [7]. Starting with a small number of interactively identified seed points within the aorta, the initial surface Γ propagates in an outward direction with the speed F. The fast marching segmentation algorithm stops the surface in the vicinity of object boundaries yielding an approximate object surface.

In order to achieve an accurate segmentation, a skeletonization algorithm [8] is applied to the result of the approximate segmentation to extract the aortic centerline. As a last segmentation step, a cylindrical surface graph search method is used to accurately determine the final luminal surface.

Accurate aortic surface segmentation Optimal surface detection [9] is an efficient segmentation algorithm applicable to tubular surfaces such as blood vessel. The method consists of 1) a coordinate transformation, 2) surface detection using dynamic programming, 3) mapping of the segmentation result back onto the original image. This method has been utilized in the reported work.

2.2 Point Distribution Model

A Point Distribution Model (PDM) was built on which independent components suitable for discrimination were estimated.

The Point Distribution Model of the aorta population was obtained using the segmentation results. Building the PDM consists of two stages: 1) Automatic generation of aortic landmarks on the 3D segmentation result, using a generated template shape with landmarks and a subsequent landmark mapping. A marching cubes algorithm [10] was used to generate triangular meshes, and vertices of these triangular meshes were used as landmarks.. 2) Capturing the shape variation by performing independent component analysis on the shape vectors of the individual aortic instances. After landmarking each resulting shape sample was represented by a shape vector $x = (x_1, y_1, z_1, ..., x_m, y_m, z_m)$, consisting of m sets of (x, y, z) coordinates of the landmark points.

2.3 Independent component analysis

The ICA approach in this study was based on the assumption that the observed signal vector x can be described as a vector of n linear mixtures of p independent non-gaussian source signals represented by the random vector s. The signal vector contains the landmarks of an instance of the aorta. The source signals are assumed centered and of unit variance. The mixture process, performed by the unknown mixing matrix $A_{n\times p}$ is governed by

$$x = A \cdot s \tag{1}$$

To calculate the original sources a de-mixing matrix $W_{p\times n}$ is introduced by the following equation.

$$y = W \cdot x \tag{2}$$

Estimating the de-mixing matrix W is done by maximizing the belief that the estimated sources Y are independent sources. The different existing methods find projections that maximize some independence measure of the distributions of the estimated sources. The central limit theorem states that a mixture of signals is more gaussian distributed than the individual parts. The original sources s can be recovered except for a scaling factor, if the number of observed signals n are at least as big as the number of sources p.

Kurtosis The Kurtosis K of the distribution of a random variable is one measure of Gaussianity. Kurtosis is included it in this paper because it has some simple analytical properties. The Kurtosis is defined by

$$K(x) = \frac{\mathrm{E}\{x^4\}}{\mathrm{E}\{x^2\}^2} - 3, \tag{3}$$

where x is a random variable. It can be shown that the Kurtosis is 0 for a Gaussian distribution. For practical estimation Kurtosis is far from the optimal measure due to sensitivity to outliers [11]. For theoretical considerations this does not pose a problem, and for two random independent variables x and y it holds that

$$K(x+y) = K(x) + K(y), \tag{4}$$
$$K(cx) = c^4 K(x), \tag{5}$$

where c is an arbitrary constant. Let the row vector w be a projection wx on the input data x, and let the projection vector be bounded by $\mathrm{E}\{(wx)^2\} = 1$. As stated earlier x is assumed to be generated by the model $x = As$ (1). Let z be defined by $z = wA$ and observe that $\mathrm{E}\{(wx)^2\} = wA\mathrm{E}\{s^2\}(wA)^T = \|z\|^2 = 1$, since the sources are independent and assumed of unit variance.

$$K(wx) = K(wAs) = K(zs) = \sum_{i=1}^{p} z_i^4 K(s_i). \tag{6}$$

To find distributions diverging from the Gaussian distribution, the numerical value of the Kurtosis can be maximized under the constraint $\|z\|^2 = 1$. This can be shown to be the canonical base vectors $\pm e_i$, projections on only one independent source. Intuitively, remembering the constraint $\|z\|^2 = 1$, it is also expected that maximizing Kurtosis corresponds to distributing the variance over fewer components, as values smaller than one raised to the power of four are reduced even more.

The number of source signals Maximizing the absolute value of the Kurtosis can be interpreted as recovering a projection, that is only directed along a single of several independent components. Now examining $w^T x = w^T As$ the normal assumption in ICA is that $A_{n\times p}$ satisfies $n >= p$ because in this way no constraints are imposed on z given by $z = wA$. Assuming that $n < p$ gives that

wA is only spanning a subspace of $\mathbb{R}^n$, the space of z. This could mean that some of the minima are not described in this subspace. Denote the subspace of z's space not spanned by wA by $\hat{V}_{p-n \times p}$. The additional constraints on z are given by (7), where $0_{1\times p-n}$ is a vector of zeros due to the orthogonality.

$$z\hat{V}^T = 0_{1\times p-n} \tag{7}$$

The number of constraints under the maximization is bigger than the number of parameters and thus the earlier described minima can not be reached. The Kurtosis measure is still favoring distributing the z_i's on as few components as possible though, and though recovering a true independent component is not to be expected, the maxima will by this measure be more independent than an arbitrary linear mixture of the source signals.

To illustrate the properties of maximizing the Kurtosis, an example of a randomly selected mixing matrix $A_{2\times 3}$ is chosen. This corresponds to 3 sources but only two observables. The Kurtosis of the three distributions are also randomly chosen (8).

$$A = \begin{bmatrix} 0.6136 & 1.0320 & 0.7604 \\ -0.8242 & -0.4344 & 1.2546 \end{bmatrix} \quad K_1 = 0.118 \quad K_2 = 0.7005 \quad K_3 = 2.133. \tag{8}$$

The projection vector w is rotated from 0 to π and the size is set to match the constraint $\mathrm{E}\{(wx)^2\} = 1$, z is still defined by $z = wA$. The result is seen in Fig. 2. The rotation of w giving the maximum Kurtosis is seen to include mainly one of the three independent components, whereas the two eigenvectors, defined by the maximum and the minimum of the dash-dotted curve, are mixtures of comparable fractions of all three independent components. This illustrates the tendency, that the Kurtosis measure under constraints as without constraints is better than the PCA measure at isolating a few independent components.

The algorithm For recovering the independent components the *FastICA* algorithm is applied due to its fast convergence and robustness [11]. As mentioned in Sect. 2.3 the Kurtosis is not very well suited in practical implementations with only a limited number of samples. The *FastICA* algorithm is iterative and finds the components sequentially. The weight vector w is randomly initialized which influences the obtained solution due to multiple local maxima. Several w's were initialized in order to be able to select the one giving the source with the most desirable properties, namely the best separation between diseased and normal subjects. The multiple initialization scheme is crucial in finding components well suited for discrimination.

Ordering measures Two different ordering measures are introduced in this paper. An ordering measure for extraction of the component that separates the diseased and normals, and an ordering measure that maximizes localization, which is preferable in the interpretation of the extracted components.

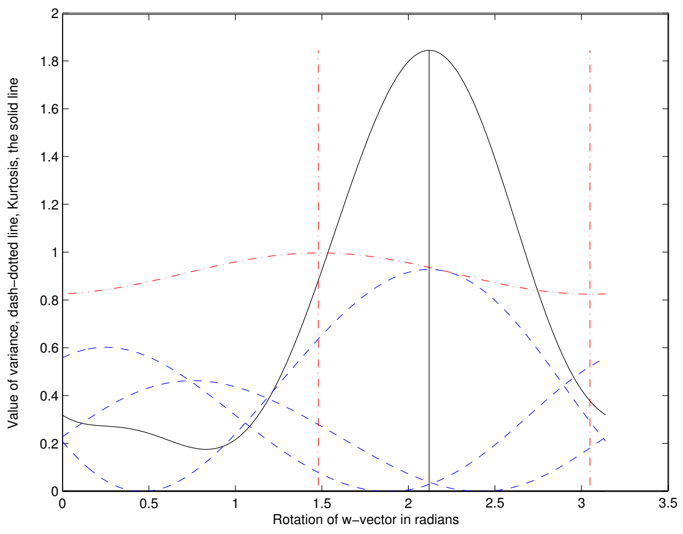


Fig. 2. w-projections in an over-constrained independent component system. The x-axis is the rotation of w in radians. The solid line is the calculated Kurtosis with the maximum illustrated. The dashed lines are representing the fraction of variance contributed from each independent component. The dash-dotted line is the variance of the projection along the w-direction.

The Fisher discriminant The hypothesis of this study is that connective tissue disorder is one of the sources shaping the aorta. Having no exact knowledge of the distribution of such a component, it is modeled to be composed of two normal distributions, one representing the normals and one the diseased offset by the difference between being diseased and having a normal aorta. As an ordering measure the Fisher discriminant, evaluating the projection separation the two populations, is expected to have its maximum at the true source.

The localization of the components The true sources are believed to be localized in the sense that the effect of being diseased for instance is expected not to influence the entire shape of the aorta, but only a part of it close to the heart. A measure is defined, that focuses on the peaks of the variance, extending a measure defined in [12] to 3D. The variation of the shape by a given projection is mapped onto the normals of the mean surface. Peaks with a peak value of over 50% of the maximum peak value are counted and the average volume of these peaks taken as a measure of how the component has centered its changes to the shape in these large peaks. The principal components represent global variations which can be seen in Fig. 3.

Independent component analysis on the aorta The data is describing the shape of the aorta and therefore the number of independent source signals is

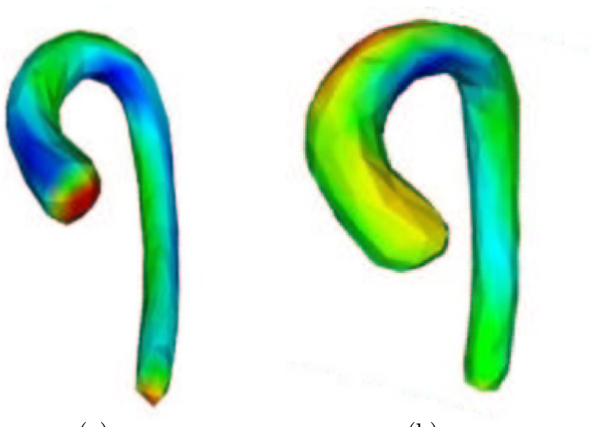


Fig. 3. (a) Aortic shape variations captured by the first PCA mode. (b) Shape variations captured by the second PCA mode. Variance from the mean shape is represented by colors from blue to red. Notice the big variance in color over the whole aortic surface.

expected to be rather high. The physical shape of the subject, the gender of the subject, the height and the age of the subject just to mention a few, could all be independent sources shaping the aorta. One of our interest divides the subjects in two groups with or without connective tissue disorder.

The number of dimensions is an important factor because the data is very sparse. The observed data X in this study have $n_{landmarks} \cdot 3 = 248 \cdot 3 = 744$ dimensions when using one phase and 1581 when using two phases of the cardiac cycle. The number of samples is only 31, 21 normals and 10 diseased. For computational convenience and because the data is only distributed along these directions, the data is projected onto the principal components. To reduce the number of free parameters a constraint is introduced, that only a certain number of the parameters are allowed to change at a time, during the estimation of the independent components. 22 principal components were retained explaining 97.5 % of the variance. The subspace was divided in 5 ($\approx \sqrt{22}$) subspaces each containing $\approx 20\%$ of the variance. Each subspace was initially searched for independent components. Subsequently independent combinations of the found projections were found and combined to form the actual independent components. This approach proved necessary to avoid overfitting due to the high number of free paramters compared to the number of samples. In this work this way of constraining different elements is found to give much more robust results than just reducing the search space by only retaining a few principal components. Setting the 16 least-variance principal components to zero, were originally chosen as the constraints on the ICA, which gave a good separation, but a poor ability to generalize.

Due to the reduced number of free dimensions and complex shaping of an aorta there are probably more sources than dimensions of the observed signal. Several locally stable projections are found, depending on the initialization of the algorithm, and it is assumed, based on the discussion in Sect. 2.3, that the different projections favor different source signals. None of them may fully describe a true source signal, but it will be more or less represented in every projection. This is the motivation for choosing an ordering measure that favors the components that is believed to describe the sources well. The aortic shape of each subject after application of ICA is represented by the projection on the independent components. As the components are chosen with the property to divide the two populations, ICA is applied again on the (two) most significant projections to extract more localized components, because we a priori believe the sources are localized.

2.4 Discrimination model

The disease detection is based on the scores of the data projected on the independent components. A simple quadratic classifier is employed on the scores of the two most significant independent components. This simple form is reinforcing that the components can be interpreted in clinical terms which has been a strong motivation for using ICA.

3 RESULTS

3.1 Segmentation result

3D candy-cane view and outflow tract MR images were acquired and merged to form a 3D image at the R-wave peak and at the time point of 45% R-R interval from 31 subjects (21 normal, 10 diseased) with image resolution ranging from $1.5 \times 1.5 \times 3.0$ mm^3 to $2.0 \times 2.0 \times 6.0$ mm^3. To assess the accuracy of the automated 3D segmentation, the aortic surfaces were compared with the expert tracing outlines. The positioning errors were defined as the shortest distances between the manually traced surfaces and the computer-determined surfaces in the 3D aortic images.

The developed segmentation method produced aortic surfaces with subvoxel accuracy as judged by the signed surface positioning errors of -0.09±1.21 voxel (-0.15 ± 2.11 mm) and unsigned positioning errors of 0.93±0.76 voxel (1.62 ± 1.25 mm). An example of a typical segmentation result is shown in Fig. 4. The segmentation result is shown in transverse and coronal views. For each view shown in the figure, 4 slices were randomly selected from the 3D image. The volumetric representation of segmentation is shown in Fig. 5.

3.2 Disease detection results

To assess the performance of the diagnostic model, two different classification tasks were performed: 1) Disease status prediction using features generated from

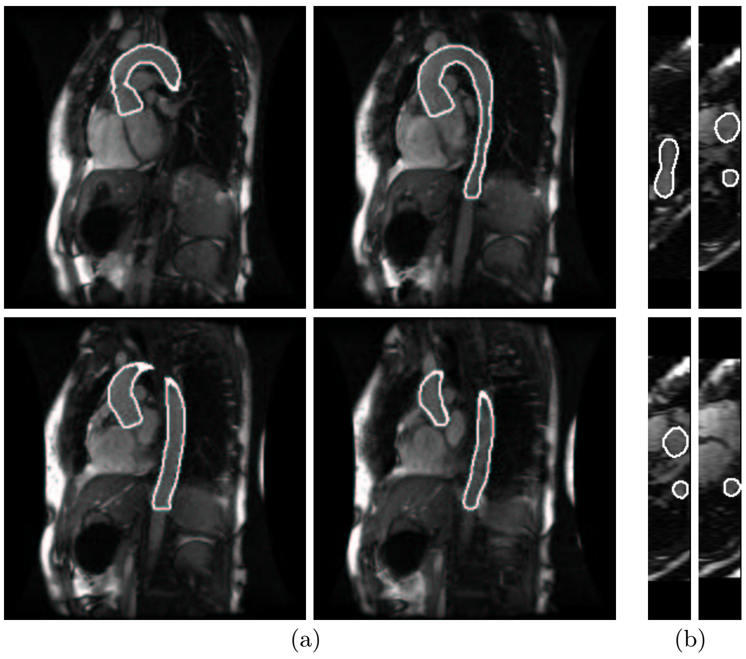


Fig. 4. Automated segmentation result in 4 randomly selected slices; the segmentation outlines are shown in green. (a) Transverse view. (b) Coronal view.

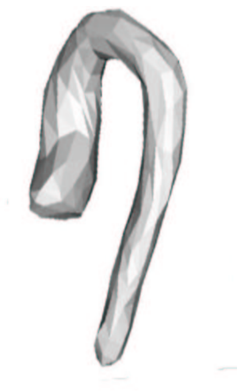


Fig. 5. Volumetric representation of the segmentation result.

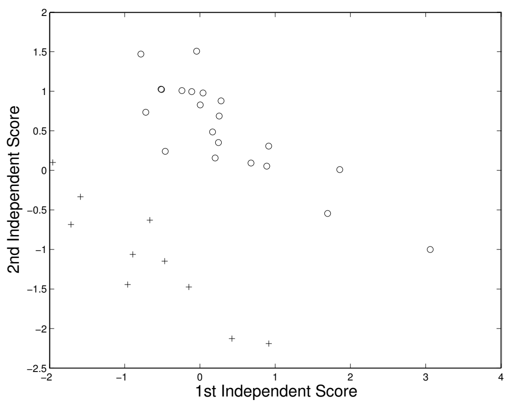


Fig. 6. Projection of data along the two first independent components. Diseased subjects are marked with '+' and healthy subjects with 'o'. A clear separation is observed.

single-phase MR images. 2) Disease status prediction using features generated from two cardiac phases. Features generated by the ICA were used as input while expert-defined disease status formed the binary prediction output (normal/diseased). A leave-one-out validation method was used to evaluate the predictive classifier performance. Performance was assessed in terms of the sensitivity, the fraction of correctly identified diseased and the specificity, the fraction of correctly classified normals.

Figure 7 illustrates the shape variations captured by the first and second independent components applying ICA on the first phase. The analysis suggests that the independent components mainly represents the variation along the aortic arch and the ascending aorta. Together they describe a dilation for diseased subjects which corresponds to the clinical observation that the effect is centered around the ascending aorta. The distribution of the projection of the data on the two first independent components, Fig. 6, illustrates that the separation task can be performed by a simple classifier. Though it always seems possible to find estimates of independent components dividing the two populations, it is not guaranteed to generalize to the unseen sample.

The two first independent components for the second phase of the aortic cycle are more localized, but show similar results.

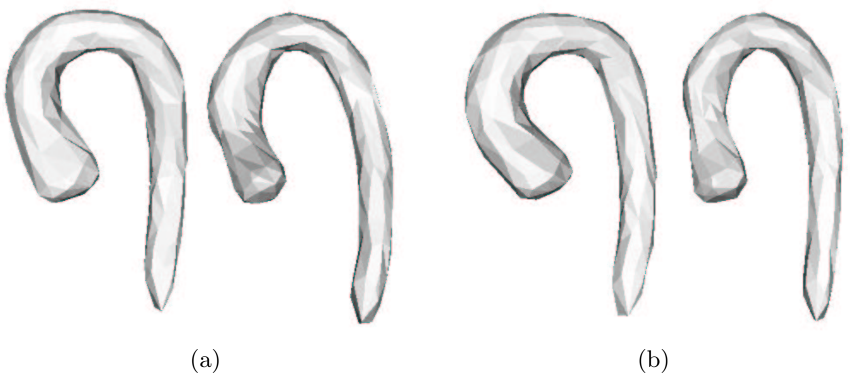


Fig. 7. Aortic shape variations observed in the analyzed population. The projection of the independent components on the normal of the mean shape surface is projected onto the mean aorta shape. Left corresponds to negative and right to positive variation. Positive values of the projection correspond to higher likelihood of being healthy. The diseased subjects (left) have a thicker aortic arch and a dilated ascending aorta. (a) Shape variations for the first independent component. (b) Shape variations for the second component.

For the single-phase case, 248 landmarks were automatically generated on each aortic luminal surface. The quadratic classifier working on two independent components exhibited a sensitivity of 80%, meaning that 80% of diseased were diagnosed as such and a specificity of 100%, meaning that all normal subjects where classified as being normal in the leave-one-out test. The classification proved worse when using the model on two phases, a sensitivity of only 70%.The localization ordering measure was designed for only one object and not two phases and this may have affected the outcome, but tests on more subjects are required to see if this difference is significant.

The overall results are summarized in Table 1 and Table 2, showing the confusion matrices of the single-phase model and the two-phase model.

The single-phase model applied to either one of the two phases gives the same confusion table, but one of the errors in classifying the diseased was for different subjects, so a combination of the one phase models might actually give an even better classification. The very encouraging results obtained analyzing a single phase is the motivation for further exploration analyzing 2 phases and later also 16 phases. An issue that may also make the sensitivity worse than the specificity could be that the number of diseased is only 10 compared to 21 normals. Though more laborious, ICA can also be a more specific tool than PCA, when reducing dimensionality as only two components are needed to feature the classification. With a prior knowledge of desired features of the component, more task-specific information can be contained in the independent components than in the variance ordered principal components by applying a suitable ordering measure.

	Predicted	
Disease Status	Diseased	Normal
Diseased	8	2
Normal	0	21

Table 1. Classification results of the single-phase model

	Predicted	
Disease Status	Diseased	Normal
Diseased	7	3
Normal	0	21

Table 2. Classification results of the two-phase model.

4 DISCUSSION AND CONCLUSION

In this study, a computer-aided diagnostic method using ICA to identify subjects with connective tissue disorders from 3D aortic MR images was presented. Accurate and reliable aortic surfaces were provided by an automated 3D segmentation algorithm which combines a fast marching level set segmentation with an optimal graph-based border detection.

Independent component analysis in a high-dimensional space with sparse data was applied to landmarked 3D shapes resulting from the aortic segmentation and formed a very efficient approach for capturing the structure's shape variation important to the classification task. A simple quadratic classifier was efficient for the simple classification task in the 2D space spanned by the two first independent components. ICA is well suited for assisting in the disease diagnostic task at hand, as it assists in an easily interpretable classification. It is shown that ICA is a well suited tool for dimensionality reduction, when prior information about the desired features exist for ordering the components. Two examples of ordering measures are introduced.

Using 3D MR image data of a single cardiac phase per subject, the classification accuracy using a basis of two independent components was 93.5%. The independent components showed a more localized behavior than the principal components and could be easier interpreted in terms of shape variation. Earlier results using a support vector machine for classification demonstrated the importance of utilizing functional information about the aortic motion for the connective tissue disease diagnosis using shape modeling, and our continued effort will be put into benifitt from the second available cardiac phase. For improved classification accuracy we will also explore the utility of full 4D information (3D + 16 cardiac phases) in the near future.

References

1. *The Centers for Disease Control and Prevention (CDC).* http://www.cdc.gov/.
2. D. Rueckert, P. Burger, S. Forbat, R. Mohiaddin, and G. Yang, "Automatic tracking of the aorta in cardiovascular MR images using deformable models," *IEEE Trans. on Medical Imaging* **16(5)**, pp. 581–590, 1997.
3. T. Behrens, K. Rohr, and H. Stiehl, "Robust segmentation of tubular structures in 3D medical images by parametric object detection and tracking," *IEEE Trans. on Systems, Man, and Cybernetics* **33(4)**, pp. 554–561, 2003.
4. M. de Bruijne, B. van Ginneken, M. A. Viergever, and W. J. Niessen, "Adapting active shape models for 3D segmentation of tubular structures in medical images," in *Proceedings of IPMI'03*, pp. 136–147, 2003.
5. M. Subasic, S. Loncaric, and E. Sorantin, "3D image analysis of abdominal aortic aneurysm," in *Proceedings of SPIE'02*, pp. 1681–1689, 2002.
6. F. Zhao, H. Zhang, N. E. Walker, F. Yang, M. E. Olszewski, A. Wahle, T. Scholz, and M. Sonka, "Quantitative analysis of two-phase 3D+time aortic MR images," *SPIE 2006* **Image Processing**, p. not published yet, 2006.
7. R. Malladi and J. A. Sethian, "A real-time algorithm for medical shape recovery," in *Proceedings of ICCV '98*, pp. 304–310, 1998.
8. K. Palagyi, E. Sorantin, E. Balogh, A. Kuba, C. Halmai, B. Erdhelyi, and K. Hausegger, "A sequential 3D thinning algorithm and its medical applications," in *Proceedings of IPMI'01*, pp. 409 – 415, 2001.
9. K. Li, X. Wu, D. Z.Chen, and M. Sonka, "Efficient optimal surface detection: Theory,implementation and experimental validation," *Proceddings of SPIE'04* **5370(48)**, pp. 620–627, 2004.
10. W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," in *Proceedings of SIGGRAPH '87*, **21(4)**, pp. 163–169, 1987.
11. A. Hyvärinen, "Survey on independent component analysis," *www.cis.hut.fi/ aapo/* **1**, pp. 1–35, 2001.
12. M. Üzümcü, A. F. Frangi, J. H. Reiber, and B. P. Lelieveldt, "Independent component analysis in statistical shape models," *Proceeding of SPIE* **5032**, pp. 375–383, 2003.

Optimizing Fast Motion Estimation and Complexity Control in H.264/AVC

Mo Wu, Kamran Virk, Mikkel Munch Christensen and Søren Forchhammer

Research Center COM, B.345V, DTU, Lyngby, Denmark

ABSTRACT

A fast motion estimation for inter motion search is provided for H.264/AVC codec down to 8×8 block size. The computational complexity is reduced to around one third of the fast algorithm of the reference software and the rate-distortion performance is maintained. A complexity control algorithm for interlaced video with $IBBP_{(12)}$ GOP structure is also proposed. The algorithm can control the complexity with a given target complexity and the Rate-Distortion-Complexity performance is improved by a complexity prediction model and control scheme. The algorithm also works well for scene change conditions.

Keywords: H.264, Fast motion estimation, complexity, R-D-C.

1. INTRODUCTION

The H.264/MPEG-4 part 10 AVC hybrid video coding standard[12] is the new video coding standard. The encoding performance is improved by a factor of 2 compared with the MPEG-4/part 2 standard. The gain in coding efficiency comes at a price of a significant increase in encoding complexity. The main part of the increased complexity stems from new features in the inter motion estimation stage, which can support seven different block sizes (16x16, 16x8, 8x16, 8x8, 8x4, 4x8 and 4x4) and multiple reference frames. Simple Full-Search motion estimation on all combinations of different block sizes and different reference frames is not feasible for real-time applications.

The normal solution of limited computation is fast motion estimation. The Enhanced Predictive Zonal Search (EPZS) fast motion estimation algorithm[3] of integer inter motion estimation was implemented with some modifications for the H.264 reference software encoder in a previous work.[5] It is viewed as a good fast motion estimation solution for the H.264/AVC codec. The 3-D recursive search (3DRS)[9] algorithm is a recursive search algorithm which is based on the philosophy of phase-plane correlation. A match error criterion is thereby able to only select of a limited number of candidate vectors.

The traditional video coding performance is measured by Rate-Distortion (R-D) performance. When complexity is also considered as an important factor here, the performance is measured by Rate-Distortion-Complexity (R-D-C). With the limited computation power and limited bandwidth, a less complex solution must be applied to real-time implementation and it should not decrease the R-D performance a lot at the same time. A solution has been proposed to control the complexity by increasing or decreasing the percentage of skipped macroblocks with the consideration of a Lagrangian R-D-C cost function.[7]

An operational method for measuring the selection of the macroblock (MB) mode and number of reference frames optimally has been developed.[5] The basic idea is to transform the three-dimensional problem of concurrently optimizing Rate-Distortion-Complexity into a more tractable two-dimensional problem. Using fixed quantization initially in the experiments, the minor changes in distortion due to the complexity control can be converted into small changes in rate by using a local slope of the R-D curve for the searched parameter setting. This effectively eliminates the distortion parameter, and the optimization problem has thus been reduced to a problem between two parameters, rate* (R^*) and complexity (C).

In the following paper, a modified 3DRS algorithm will be proposed for variable block sizes in Section 2; the complexity control algorithm based on EPZS is shown in Section 3; then the results and conclusion are illustrated at last.

Further author information: (Send correspondence to Mo Wu.)
Mo Wu: E-mail: mw@com.dtu.dk, Telephone: +45 4525 3620

2. MODIFIED 3DRS FOR VARIABLE BLOCK SIZES

The proposed algorithm is a modified 3DRS algorithm. It utilizes variable block size selection in order to improve the computational load of real-time H.264. The basic outline of the algorithm can be seen in Figure 1.

The idea in the structure of this VBS-3DRS algorithm is the following:

- All modes are utilized in 3DRS fashion
- At initialization zero motion vector (MV) and one pel refinement helps 3DRS to utilize spatial candidates. Temporal candidates are utilized after the very first frame.
- SAD (summed absolute difference) is calculated for motion vectors. The total cost is composed of SAD plus motion vector cost. If the cost is lower than the adaptive threshold then the rest of the algorithm is skipped and if higher the next step in the algorithm is performed.
- 3DRS is performed. If the new cost is lower than the threshold after this step, one pel integer search is skipped otherwise it is performed in order to further improve the cost and thereby get a better candidate motion vector with the respective mode.

2.1. VBS-3DRS

At initialization the motion vectors and best modes for different blocks are decided through one pel search. As soon as 3DRS spatial candidates (B(X), C_{S1}, C_{S2}) are available, spatial prediction is performed. Where B(x) is the current block for which motion estimation is being done. The modes and motion vectors in the first frame are stored to be used in the next frame. From the second frame, the 3DRS algorithm utilize temporal candidates, C_{T1} also. After having spatial and temporal prediction, best motion vectors is collected along with associated modes. Also from stated theory of 3DRS algorithm, this combination of the spatial and temporal prediction provides an advantage in giving a look ahead into the direction of motion.

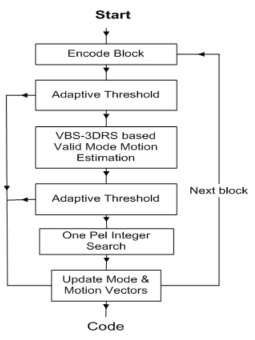


Figure 1. VBS-3DRS

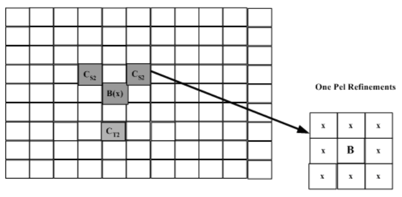


Figure 2. Enhanced 3DRS candidate placement and one pel refinement around the best match (B)

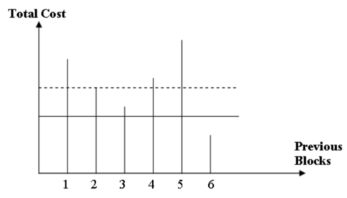


Figure 3. Adaptive mean threshold

Best motion vectors are selected from SAD based cost which is calculated for each candidate using the candidate block motion vectors. If the SAD represents the cost of a candidate and min cost is the minimum cost of the four blocks against which comparisons are made then from the candidates we have:

$$SAD = min\Big(SAD(C_{S1}), SAD(C_{S2}), SAD(C_{T1}), SAD(B(x))\Big) \tag{1}$$

$$TotalCost = SAD + \lambda \cdot MVCost \tag{2}$$

where MVCost is the cost for the bits required to code the motion vector, and *lambda* is a Lagrangian multiplier decided by the reference codec.

The best mode is then the mode, for which equation (2) gives the gives the smallest cost. The best motion vectors are passed on to the next stage for further refinement.

2.2. One pel integer search

In order to improve the results from the VBS-3DRS a refinement of the best match is performed. Before refinements, the best VBS-3DRS cost is compared with an adaptive mean threshold (see below) and it allows refinements performed over best match (B) as shown in Figure 2. The refinement can be skipped when the cost is compared with the threshold as shown in Figure 1. If a better match is found then the best motion vectors along with mode information is updated accordingly.

2.3. Adaptive mean threshold

The idea of the adaptive mean threshold is to make an adaptive threshold which is very simple, hence gaining flexibility without introducing computational complexity. The threshold is constructed by taking the minimum cost of the previous 6 blocks. The highest and the lowest cost is subtracted and a mean of the rest is found. As shown by the punctured line in Figure 3 in order to get a cost for the current frame which is better than the mean cost, 1/4 is subtracted from the mean, indicated by a solid horizontal line. By doing this we will only accept costs better than the mean if possible.

3. COMPLEXITY CONTROL OF FAST MOTION ESTIMATION IN H.264/MPEG-4 AVC WITH RATE-DISTORTION-COMPLEXITY OPTIMIZATION BASED ON EPZS

A complexity control algorithm is designed based on the previous extended EPZS fast motion estimation work.[5] The fast motion estimation scheme is reserved and some other control related algorithm is added to the original work.

3.1. Counting method of the inter search complexity, C, and N

The motion search complexity here is measured by the weighted search positions, which are viewed as a good measurement of the motion search complexity [4]. The weighted search positions of different block partitions are counted according to the block size searched in the inter mode motion estimation (see Table 1).

Table 1. *weighted search positions of different block partitions per search*

Block partition	16×16	16×8	8×16	8×8	8×4	4×8	4×4
Weighted search positions per search	16	8	8	4	2	2	1

The total number of the weighted search positions, C, is accumulated for coding each frame using integer accuracy inter motion estimation. Inter mode search complexity is here measured by the number of *weighted search positions per second* [4] (WSP/s) in the experiment. Complexity, C, is a function of parameters in the configuration of the encoder. In this problem, C can be viewed as a function of the block partition modes, the number of reference frame and the index of the frame, i; the minimum block partitions, $b(i)$, and the number of the reference frames, $r(i)$, of one frame are a function of the frame index, i;

$$b(i) = \begin{cases} 1 & partitions : 16 \times 16, \\ 2 & partitions : 16 \times 16, 16 \times 8, 8 \times 16 \, and \, 8 \times 8, \\ 3 & partitions : all \, seven \, partitions \end{cases} \tag{3}$$

The experiment is done with interlaced video, $r(i)$ equals one when reference frames are restricted to the first reference frame (two reference fields). $r(i)$ increases by one each time one more reference field can be searched. In the configuration, the maximum number of the reference frame is five, so the maximum number of $r(i)$ is nine here. The other parameters are constant throughout the encoding process.

$$C = f(b(i), r(i), i) = \sum_i C_{PF}(b(i), r(i), i) + \sum_i C_{BF}(b(i), r(i), i), \tag{4}$$

where $C_{PF}(b(i), r(i), i)$ or $C_{BF}(b(i), r(i), i)$ is the complexity for the i_{th} frame, which is P frame or B frame respectively; $C_{PF}(b_{max}, r_{max}, i) = 0$, when frame is not a P frame; and $C_{BF}(b_{max}, r_{max}, i) = 0$, when frame is not a B frame. For H.264 video codec, except I frame, the frames are classified into two types by the properties of inter prediction performed. One is P frame and the other one is B frame. For P frame only previous frames are used as reference. While both previous and following frames are utilized for prediction for B frame.[2]

$$\begin{cases} C_{PF}(b(i), r(i), i) & = \sum_{b=1}^{b(i)} \sum_{b=1}^{b(i)} C_P(b(i), r(i), i), \\ C_{BF}(b(i), r(i), i) & = \sum_{b=1}^{b(i)} \sum_{b=1}^{b(i)} C_B(b(i), r(i), i), \\ C_P(b, r, i) & = 0, \qquad ifb > b(i), \\ C_B(b, r, i) & = 0, \qquad ifb > b(i), \\ b, b(i) & \in \{1, 2 \quad and \quad 1\}, \\ r, r(i) & \in \{m | 1 \leq m \leq 9, m \in N\} \end{cases} \tag{5}$$

$r = 1$ represents the first two reference fields, $r = 2 \sim 9$ represents the rest of the reference fields (the maximum number of reference frame is 5 here); $C_P(b, r, i)$ or $C_B(b, r, i)$ is the complexity for inter motion search of the i_{th} frame (P or B frame respectively) on the r_{th} reference and b_{th} partition modes, which are described in eq. (6).

$$b = \begin{cases} 1 \quad partitions: 16 \times 16, \\ 2 \quad partitions: 16 \times 8, 8 \times 16\, and\, 8 \times 8, \\ 3 \quad partitions: 8 \times 4, 4 \times 8\, and\, 4 \times 4 \end{cases} \tag{6}$$

N is the accumulated weighted number of 4x4 blocks compressed in inter mode. Similar to the definition of C, N apply the same weighted number for different block sizes as in Table 1. N can be viewed as a function of the block partition modes, the number of reference frames and the frame index.

$$N = g(b(i), r(i), i) = \sum_i N_{PF}(b(i), r(i), i) + \sum_i N_{BF}(b(i), r(i), i), \tag{7}$$

where $N_{PF}(b(i), r(i), i)$ or $N_{BF}(b(i), r(i), i)$ is the weighted number of 4x4 blocks for the i_{th} frame, which is respectively P frame or B frame; $N_{PF}(b_{max}, r_{max}, i) = 0$, when frame is not a P frame; and $N_{BF}(b_{max}, r_{max}, i) = 0$, when frame is not a B frame.

$$\begin{cases} N_{PF}(b(i), r(i), i) & = \sum_{b=1}^{b(i)} \sum_{b=1}^{b(i)} N_P(b(i), r(i), i), \\ N_{BF}(b(i), r(i), i) & = \sum_{b=1}^{b(i)} \sum_{b=1}^{b(i)} N_B(b(i), r(i), i), \\ N_P(b, r, i) & = 0, \qquad if\ b > b(i), \\ N_B(b, r, i) & = 0, \qquad if\ b > b(i), \\ b, b(i) & \in \{1, 2 \quad and \quad 1\}, \\ r, r(i) & \in \{m | 1 \leq m \leq 9, m \in N\} \end{cases} \tag{8}$$

where $N_P(b, r, i)$ or $N_B(b, r, i)$ is the weighted number of 4x4 blocks for inter motion search of the i_{th} frame (P or B frame respectively) on the r_{th} reference and b_{th} partition modes. Because of SKIP mode, bi-prediction etc [8] of H.264/AVC codec, the counting method of $N_P(b, r, i)$ and $N_B(b, r, i)$ are not as simple as that of $C_P(b, r, i)$ or $C_B(b, r, i)$, and they are shown as follows,

```
If (mode == 0) // skip mode for P frame or direct mode for B frame
{
   for (first 4x4 block; the last 4x4 block; next 4x4 block)
   {
      If (P frame) $N_P(1, 1, i) + +$;
```

```
        If (B frame)
        {
            If (only FW) $N_B(1,r,i)++$;
            If (only BW) $N_B(1,1,i)++$;
            If (Bi-prediction) $\{N_B(1,r,i) += \frac{1}{2}; \{N_B(1,1,i) += \frac{1}{2};\}$
        }
    }
}
If ($1 \leq mode \leq 7$) // inter mode with partitions mode $1 \sim 7$
{
    for (first 4x4 block; the last 4x4 block; next 4x4 block)
    {
        If (P frame) $N_P(b,r,i)++$;
        If (B frame)
        {
            if (only FW) $N_B(b,r,i)++$;
            if (only BW) $N_B(b,1,i)++$;
            if (Bi-prediction) $\{N_B(b,r,i) += \frac{1}{2}; N_B(b,1,i) += \frac{1}{2};\}$
        }
    }
}
```

Here in the experiment, the direct mode is tested in spatial mode. In a P frame's skip mode or a B frame's direct mode, there is no motion vectors that are coded. But the encoder and the decoder will use the same predicted MVs, constructed by previously compressed MVs. Thus, except of the compression method of motion vectors, the same motion compensation scheme is applied. It is reasonable to count them into $N_P(b,r,i)$ or $N_B(b,r,i)$ with the relevant reference frame and block type.

3.2. Complexity prediction

Through the analysis of the statistics of three test sequences, Mobcal, cycling and Barcelona, a more elaborate method is applied to predict a good setting, $b(i)$ and $r(i)$ with a given complexity. In the experiment, the GOP structure is $IBBP_{(12)}$. Inside each GOP, there are three BBP_s, which are called P-GOP$_s$ in the following content. The complexity prediction is applied on the complexity of P-GOP. The complexity of each P-GOP frames can be defined as,

$$\begin{cases} C_{BBP}(b(i),r(i),b(i+2),r(i+2),i) \\ = \sum_{j=i}^{i+1} C_{BF}(b(j),r(j),j) + C_{PF}(b(i+2),r(i+2),i+2), \\ b(i) = b(i+1), \\ r(i) = r(i+1) \end{cases} \tag{9}$$

The block partition settings can be different for P and B frames, whereas the two B frames have the same setting. For the first ($IBBP$), block searching type is set to only 16x16 partition mode and can only be one reference frame for the first P-GOP. According to Eq. 9, the complexity can be described as,

$$C_{BBP}(1,1,1,1,i) = \sum_{j=i}^{i+1} C_{BF}(1,1,j) + C_{PF}(1,1,i+2), \tag{10}$$

The prediction of the complexity of different settings in the next P-GOP is expressed as follows,

$$f_{P1}(b,r,i') = \frac{\hat{C}_{PF}(b,r,i'+2) - C_{PF}(1,1,i+2)}{C_{BBP}(1,1,1,1,i)} = \begin{cases} k_0(r-1), & if\, b=1, \\ k_1(r-1)+k_2, & if\, b=2, \\ k_3(r-1)+k_4, & if\, b=3, \end{cases} \tag{11}$$

$$f_{B2}(b',r',i') = \frac{\sum_{j=i'}^{i'+1}\hat{C}_{BF}(b',r',j) - \sum_{j=i'}^{i'+1}C_{BF}(1,1,j)}{C_{BBP}(1,1,1,1,i)} = \begin{cases} k_5(r'-1), & if\, b'=1, \\ k_6(r'-1)+k_7, & if\, b'=2, \\ k_8(r'-1)+k_9, & if\, b'=3, \end{cases} \tag{12}$$

where the parameters k_m are acquired by collecting the results of

$$\frac{\sum_i C_{PF}(b,r,i) + \sum_i C_{BF}(b',r',i)}{\sum_i C_{BBP}(1,1,1,1,i)}, \tag{13}$$

which equals to

$$\frac{E[C_{PF}(b,r,i)] + 2\cdot E[C_{BF}(b',r',i)]}{E[C_{BBP}(1,1,1,1,i)]}, \tag{14}$$

Where $E[\cdot]$ denotes expectation. The statistics are fitted to the model to get parameters. $k_0 = 0.168$, $k_1 = 0.56$, $k_2 = 0.6$, $k_3 = 0.82$, $k_4 = 1.05$, $k_5 = 0.4$, $k_6 = 1.34$, $k_7 = 2.0$, $k_8 = 2.0$ and $k_9 = 3.6$; So, drawn from Eq. (11), (12), the prediction of the complexity in different settings can be calculated by the following equation.

$$\begin{aligned} \hat{C}_{BBP}(b,r,b',r',i') &= \hat{C}_{PF}(b,r,i'+2) + \sum_{j=i'}^{i'+1}\hat{C}_{BF}(b',r',i) \\ &= (f_{P1}(b,r,i') + f_{B2}(b',r',i') + 1)\cdot C_{BBP}(1,1,1,1,i), \end{aligned} \tag{15}$$

With the given P-GOP complexity, $C_{BBP}(1,1,1,1,i)$, the prediction of P-GOP complexity, $\hat{C}_{BBP}(b,r,b',r',i')$ can be predicted from Eq. (16). Then the appropriate setting could be selected from the settings using similar complexity with the target complexity.

Similarly, $\hat{C}_{BBP}(b,r,b',r',i')$ can also be predicted by a given previous P-GOP complexity, $C_{BBP}(b_0,r_0,b'_0,r'_0,i)$, which could be any possible setting.

$$\hat{C}_{BBP}(b,r,b',r',i') = \frac{(f_{P1}(b,r,i') + f_{B2}(b',r',i') + 1)}{(f_{P1}(b_0,r_0,i') + f_{B2}(b'_0,r'_0,i') + 1)}\cdot C_{BBP}(b_0,r_0,b'_0,r'_0,i), \tag{16}$$

3.3. Definition of the *benefits*

The complexity is controlled differently for the P frames and B frames. Normally, P frames consume less complexity than B frames, nevertheless it is more important for motion estimation and compensation. The setting for P and B frame is decided by the *benefits* we defined as follow,

$$B_{Pb}(b,i) = s\cdot bits_{PF}\cdot\frac{\sum_r N_P(b,r,i)}{\sum_r C_P(b,r,i)} \tag{17}$$

$$B_{Pr}(r,i) = s\cdot bits_{PF}\cdot\frac{\sum_b N_P(b,r,i)}{\sum_b C_P(b,r,i)} \tag{18}$$

$$B_{Bb}(b,i) = bits_{BF}\cdot\frac{\sum_r N_B(b,r,i)}{\sum_r C_B(b,r,i)} \tag{19}$$

$$B_{Br}(r,i) = bits_{BF}\cdot\frac{\sum_r N_B(b,r,i)}{\sum_b C_B(b,r,i)} \tag{20}$$

where $bits_{PF}$ or $bits_{BF}$ are the number of bits used in coding the P or B frames respectively; $s = 2$ is used in the experiment; *Benefits* here can be viewed as comparable terms for P and B frames with specific partition modes of b or reference of r. For example, $\frac{\sum_r N_P(b,r,i)}{\sum_r C_P(b,r,i)}$ can be viewed as the number of weighted 4×4 blocks per complexity that can be improved in the partition modes of b; s and $bits_{PF}$ are the weights from frame type and the number of bits respectively. The model of the benefits is constructed based on experiments and related to the improvement of the $R-D$ (R^*) per complexity.

3.4. Controlling process

Over a short time period, the properties of the sequence are quite similar. It is not a good solution to tune the complexity quite frequently with respect to $R-D-C$ performance. It is a better way to use complexity prediction and make the complexity control process robust to the small fluctuations of complexity and avoid to change complexity dramatically.

State 0 normal control state after the P-GOP.

State 1 just after detecting a scene change, set setting to $b = b' = r = r' = 1$; Go to state 2.

State 2 this is a state after coding first P-GOP or after state 1; If just after coding first P-GOP, set the setting by complexity prediction; If after state 1, set setting to $b = b' = 3, r = r' = 2$; Go to state 3.

State 3 Switch between the block partition priority or reference priority preset orders by comparing $\frac{\sum_{b_s=1}^{b} N_P(b_s,1,i)}{\sum_{b_s=1}^{b} N_P(b_s,1,i)}$ with threshold, T; Set setting by complexity prediction; Go to state 0.

In state 0, the control scheme is described as follow,

```
If (full fill predicted setting) // in the first several $P$-GOPs, the setting set by prediction may not be full filled, because of limited reference frames.
{
    if ($C_{P-GOP} > C_{P-GOPT}$)
    {
        while ($C'_{P-GOP} + C_{P-GOP} > 2 \cdot C_{P-GOPT}$) // $C'_{P-GOP}$ is the predicted complexity if switch off one setting
        {
        decrease the complexity by switching off search on $b_s$ or $r_s$ or $b'_s$ or $r'_s$ with $argmin\{B^P_{b_s}, B^P_{r_s}, B^B_{b'_s}\ and\ B^P_{r'_s}\}$;
        }
    }
    else
    {
        if ($\frac{|C_{GOPT} - C_{GOP}|}{C_{GOPT}} < 0.05$) {continue}; // 0.05 is a threshold to make the control robust to the fluctuation of the complexity
        else {reset the setting by complexity prediction;}
    }
}
else
{
    if ($C_{P-GOP} > C_{P-GOPT}$) { reset the setting by complexity prediction;}
    else {continue;}
}
```

In the control process, there is a target C_T, which is measured by *weighted search positions per second* (WSP/s). Here the control method applied targets at the complexity of P-GOP (C_{P-GOPT}), which has relation with the target C_T. The relation between C_T and C_{P-GOPT} is mainly affected by the frame rate and GOP structure, thus they can be viewed as having a linear relation. We can easily get

$$C_T = \frac{length\,of\,GOP - 1}{length\,of\,P - GOP} \cdot \frac{frame\,rate}{length\,of\,GOP} \cdot C_{GOPT} \tag{21}$$

With GOP structure, $IBBP_{(12)}$ and frame rate, 25 (frames/s), it is easy to get the relation, $C_T = 7.6389 \cdot C_{GOPT}$. But in reality, the relation is also slightly affected by the number of frames and C_T itself. In order to simply this problem, we assume there still exists a linear relation between these terms.

$$C_T = \alpha \cdot C_{GOPT} + \beta. \tag{22}$$

After some experiments, $\alpha = 7.223$ and $\beta = 9.7$ are chosen as the parameters of the linear relation, which is similar to the result of the simple model.

4. RESULTS

4.1. Results of the extended 3DRS

The experiment is conducted with QCIF (176×144) sequences, Foreman, Coastguard and Brea. The test sequences are tested on Pentium IV 2.6 GHz. Figure 4 shows the motion estimated frames comparison between reference software [11] and new VBS-3DRS implementations along with adaptive thresholding criteria for different standard sequences. From Figure 5, it is clear that the new implementation has better computational performance over the reference and the complexity is 32% of that of reference software in average. The relevant R^* is only increased by 3% in average.

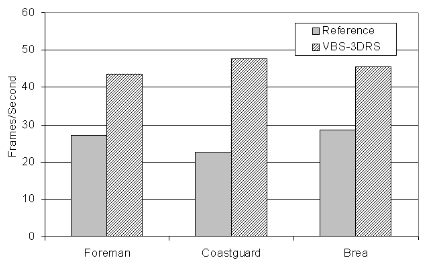


Figure 4. Number of frames motion estimated per second

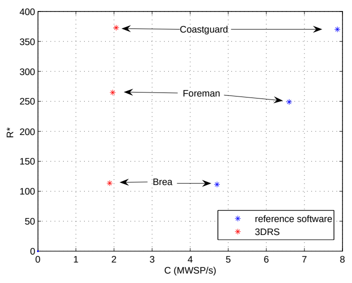


Figure 5. Performance comparison between 3DRS and reference by R*-C

4.2. Complexity control results

The adaptive complexity control method was tested on the standard interlaced PAL (interlaced SDTV) sequences. The GOP-structure is $IBBP_{(12)}$. The best average settings [4] from low to high complexity are $(1, 16)$, $(1, 8)$, $(2, 8)$, $(3, 8)$, $(4, 8)$, $(5, 8)$ and $(5, 4)$, where (r_M, b_M) represents maximum r_M reference frames and searched smallest block size down to $b_M \times b_M$.

4.2.1. Normal test sequences

The normal PAL (720x576) sequences (Mobcal, barcelona and Table) have 100 frames. The frame rate is 25 frames per second. The experimental results for the test sequences, Mobcal and Table, are shown in Figure 6. The three test sequences' average improvement of the R^* compared with best setting is 1.51%, compared with best average setting is 0.73%. The accuracy of the complexity control is shown in Table 2.

Table 2. *The difference between the target complexity and the real complexity*

Sequence	barcelona	mobcal	table
Average complexity difference (MWSP/s)	5.0966	4.2885	4.2414
$\frac{Average\ complexity\ difference}{Average\ C}$ (%)	2.41	2.54	1.96

4.2.2. Scene change

In the normal video, scene changes happen frequently. The sequences with scene change is edited by concatenating two normal 100 frames PAL sequences. The test is conducted on MobBar (Mobcal + Barcelona), MobTab (Mobcal + Table) and BarTab (Barcelona + Table). The experimental results of MobBar and BarTab are shown in the Figure 7. The three test sequences' average improvement of R^* compared with best setting is 1.7591 (MWPS/s), compared with best average setting, it is 2.0796 (MWPS/s). The average percentage of the complexity control difference to the average target complexity is 1.6733%.

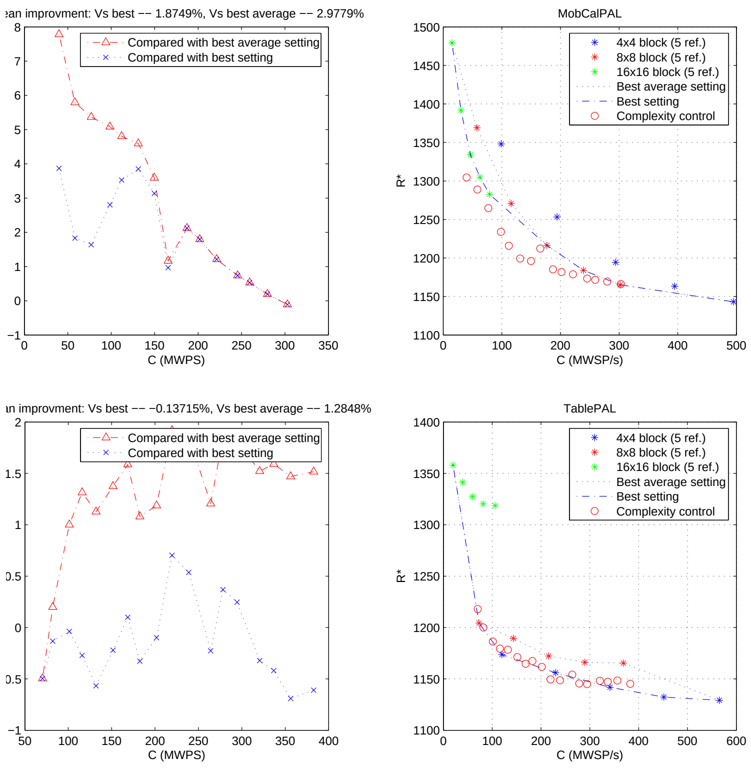


Figure 6. Test results of Mobcal and Table

Table 3. *The difference between the target complexity and the real complexity in scene change condition.*

Sequence	MobBar	MobTab	BarTab
Average complexity difference (MWSP/s)	7.3736	1.4788	3.0413
$\frac{Average\ complexity\ difference}{Average\ C}$ (%)	2.93	0.86	1.23

5. CONCLUSION

This paper presents methods for enhancing the computational performance of the H.264 encoder by employing VBS-3DRS down to 8x8 block size and one pel refinements along with adaptive mean threshold. The average SNR and bitrates of the sequences do not vary much as compared with the reference implementation. At the same time it has been shown to provide less computational load than the reference one. An effective algorithm of complexity control based on fast inter motion estimation algorithm is also provided. The algorithm can be based on different fast motion estimation algorithms, because the structure of the complexity control algorithm is separately designed from the EPZS algorithm. A new term, benefit, is defined to measure the improvement of rate distortion performance with the given computation power in the control process. The paper also gives a complexity prediction algorithm for different combinations of block sizes and number of reference frames. It applies an efficient process in complexity control with the given target of complexity whilst maintaining good rate distortion performance. Scene changes can be detected and used to govern the complexity control algorithm, leading to improved rate distortion performance.

REFERENCES

1. T. Wiegand and G. J. Sullivan (ed.), "Draft ITU-T Rec. and FDIS (ITU-T Rec. H.264-ISO/IEC 14496-10 AVC)," *H.264/MPEG-4 AVC (JVT-doc. JVT-G050)*, Mar.2003.

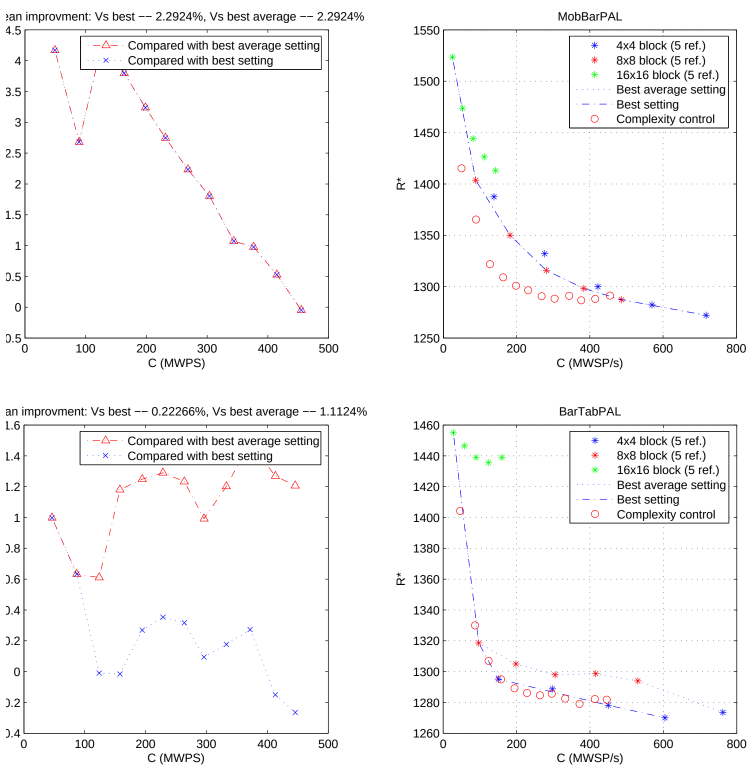


Figure 7. Test results of MobBar and BarTab

2. T. Wiegand et al.,"Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits and Systems for Video Tech.*, vol. **13**, no. **7**, pp. 1-19, July 2003.
3. A. M. Tourapis,"Enhanced predictive zonal search for single and multiple frame motion estimation," *VCIP 2002, Proc. Of SPIE.* vol. 4671, pp. 1069-1079, 2002.
4. G. Haan, P.W.A.C. Biezen, O.A. Huijgen, "True Motion Estimation with 3-D Recursive Search Block Matching", IEEE Transactions on Circuits and Systems for Video Technology, Volume3, pp. 368-379,388, Oct. 1993.
5. J. Andersen, S. Forchhammer and S. Aghito, "Rate-Distortion-Complexity optimization of Fast Motion Estimation in H.264/MPEG-4 AVC," *Proc. ICIP 2004.*
6. D. Alfonso et al., "Detailed rate-distortion analysis of H.264 video coding standard and comparison to MPEG-2/4," *VCIP 2003, Proc. Of SPIE*, vol. **5150**, pp. 891-902, 2003.
7. C. Kannangara, and I. Richardson, "Computional Control of An H.264 Encoder Through Lagrangian Cost Function Estimation", *VLBV'05*, Sardinia, Italy, Sept. 2005.
8. I. Richardson, *H.264 and MPEG-4 Video Compression*, John Wiley & Sons Ltd, 2003.
9. K. Virk, N. Khan, S. Masud, F. Nasim, S. Idris, *A Low Complexity Motion Estimation Algorithm for Video Coding Applications*, EUSIPCO-2005, 13th European Signal Processing Conference, September 4-8, 2005.
10. J. Lee, *Optimal quadtree for variable block size motion estimation*, IEEE International Conference on Image Processing, Volume 3, pp. 480 - 483, 1995.
11. *http://iphome.hhi.de/suehring/tml/download*

A Quasi-Minimal Model for Paper-Like Surfaces and its Reconstruction From Images

Mathieu Perriollat **Adrien Bartoli**

LASMEA - CNRS / UBP ◇ Clermont-Ferrand, France

Mathieu.Perriollat@lasmea.univ-bpclermont.fr, Adrien.Bartoli@gmail.com

http://comsee.univ-bpclermont.fr

Abstract

We are interested in reconstructing paper-like objects from images. These objects are modeled by developable surfaces and are mathematically well-understood. They are difficult to minimally parameterize since the number of meaningful parameters is intrinsically dependent on the actual surface.

We propose a quasi-minimal model which self-adapts its set of parameters to the actual surface. More precisely, a varying number of rules is used jointly with smoothness constraints to bend a flat mesh, generating the sought-after surface.

We propose an algorithm for fitting this model to multiple images by minimizing the point-based reprojection error. Experimental results are reported, showing that our model fits real images accurately.

1 Introduction

The behaviour of the real world depends on numerous physical phenomena. This makes general-purpose computer vision a tricky task and motivates the need for prior models of the observed structures, *e.g.* [1, 4, 8, 10]. For instance, a 3D morphable face model makes it possible to recover camera pose from a single face image [1].

This paper focuses on paper-like surfaces. More precisly, we consider paper as an unstretchable surface with everywhere vanishing Gaussian curvature. This holds if smooth deformations only occurs. This is mathematically modeled by developable surfaces, a subset of ruled surfaces. Broadly speaking, there are two modeling approaches. The first one is to describe a continous surface by partial differential equations, parametric or implicit functions. The second one is to describe a mesh representing the surface with as few parameters as possible. The number of parameters must thus adapts to the actual surface. We follow the second approach since we target at computationally cheap fitting algorithms for our model.

One of the properties of paper-like surfaces is inextensibility. This is a nonlinear constraint which is not obvious to apply to meshes, as Figure 1 illustrates. For instance, Salzmann *et al.* [10] use constant length edges to generate training meshes from which a generating basis is learnt using Principal Component Analysis. The nonlinear constraints are re-injected as a penalty in the eventual fitting cost function. The main drawback of this approach is that the model does not guarantee that the generated surface is developable.

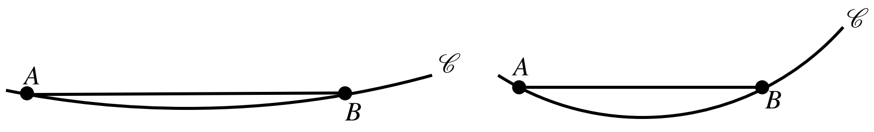


Figure 1: Inextensibility and approximation: A one dimensional example. The curve $\mathscr{C}$ represents an inextensible object, A and B are two points lying on it. Linearly approximating the arc (AB) leads to the segment AB. When $\mathscr{C}$ bowes, although the arc length (AB) remains constant, the length of the segment AB changes. A constant length edge model is thus not a valid parameterization for inextensible surfaces.

We propose a model generating a 3D mesh satisfying the above mentioned properties, namely inextensibility and vanishing Gaussian curvature at any point of the mesh. The model is based on bending a flat surface around rules together with an interpolation process leading to a smooth surface mesh. We only assume a convex object shape. The number of parameters lies very close to the minimal one. This model is suitable for image fitting applications and we describe an algorithm to recover the deformations and rigid pose of a paper-like object from multiple views.

Previous work. The concept of developable surfaces is usually chosen as the basic modeling tool. Most work uses a continuous representation of the surface [3, 4, 7, 9]. They are thus not well adapted for fast image fitting, except [4] which initializes the model parameters with a discrete system of rules. [11] constructs developable surfaces by partitioning a surface and curving each piece along a generalized cone defined by its apex and a cross-section spline. This parameterization is limited to piecewise generalized cones. [6] simulates bending and creasing of virtual paper by applying external forces on the surface. This model has a lot of parameters since external forces are defined for each vertex of the mesh. A method for undistorting paper is proposed in [8]. The generated surface is not developable due to a relaxation process that does not preserve inextensibility.

Roadmap. We present our model in §2 and its construction from multiple images in §3. Experimental results on image sequences are reported in §4. Finally, §5 gives our conclusions and some further research avenues.

2 A Quasi-Minimal Model

We present our model and its parameterization. The idea is to fold a flat mesh that we assume rectangular for sake of simplicity. We underline however that our model deals with any convex shape for the boundary.

2.1 Principle

Generating a surface mesh using our model has two main steps. First, we bend a flat mesh around 'guiding rules'. Second, we smooth its curvature using interpolated 'extra rules', as illustrated in Figure 2. The resulting mesh is piecewise planar. It is guaranteed to be admissible, in the sense that the underlying surface is developable.

Step 1: Bending with guiding rules. A ruled surface is defined by a differentiable space curve $\alpha(t)$ and a vector field $\beta(t)$, with t in some interval I, see e.g. [11]. Points on the surface are given by:

$$X(t,v) = \alpha(t) + v\beta(t) \ , \quad t \in I \quad v \in \mathbb{R} \quad \beta(t) \neq 0. \tag{1}$$

The surface is actually generated by the line pencil $(\alpha(t), \beta(t))$. This formulation is continuous.

Since our surface is represented by a mesh, we only need a discrete system of rules, at most one per vertex of the mesh. Keeping all possible rules leads to a model with a high number of parameters, most of them being redundant due to surface smoothness. In order to reduce the number of parameters, we use a subset of rules: The guiding rules. Figure 2 (left) shows the flat mesh representing the surface with the selected rules. We associate an angle to each guiding rule and bend the mesh along the guiding rules accordingly. Figure 2 (middle) shows the resulting guiding mesh. The rules are choosen such that they do not to intersect each other, which corresponds to the modeling of smooth deformations.

Step 2: Smoothing with extra rules. The second step is to smooth the guiding mesh. To this end, we hallucinate extra rules from the guiding ones, thus keeping constant the number of model parameters. This is done by interpolating the guiding rules. The folding angles are then spread between the guiding rules and the extra rules, details are given in the next section. Figure 2 (right) shows the resulting mesh.

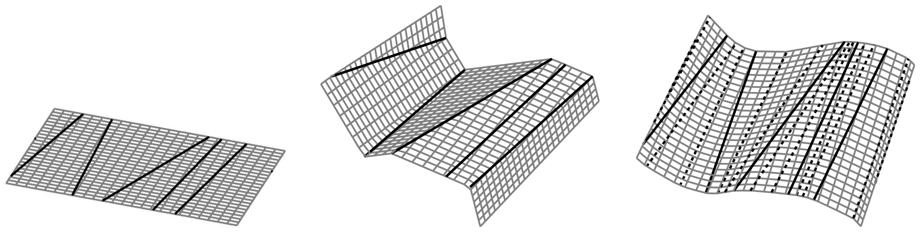


Figure 2: Surface mesh generation. (left) Flat mesh with guiding rules (in black). (middle) Mesh folded along the guiding rules. (right) Mesh folded along the guiding and extra rules.

2.2 Parameterization

A guiding rule i is defined by its two intersection points A_i and B_i with the mesh boundary. Points A_i and B_i thus have a single degree of freedom each. A minimal parameterization is their arc length along the boundary space curve. Since the rules do not intersect each other on the mesh, we define a 'starting point' P_s and an 'ending point' P_e such that all rules can be sorted from P_s to P_e, as shown on Figure 3 (left). Points A_i (resp. B_i) thus have an increasing (resp. decreasing) arc length parameter. The set of guiding rules is parameterized by two vectors s_A and s_B which contain the arc lengths of points A_i and B_i respectively. The non intersecting rules constraint is easily imposed by enforcing monotonicity on vectors s_A and s_B.

As explained above, the model is smoothed by adding extra rules. This is done by interpolating the guiding rules. Two piecewise cubic Hermite interpolating polynomials are computed from the two vectors s_A and s_B. They are called f_A and f_B. This interpolation function has the property of preserving monotonicity over ranges, as required. Figure 3 (right) shows these functions and the control points s_A and s_B. The bending angles are interpolated with a spline and rescaled to account for the increasing number of rules.

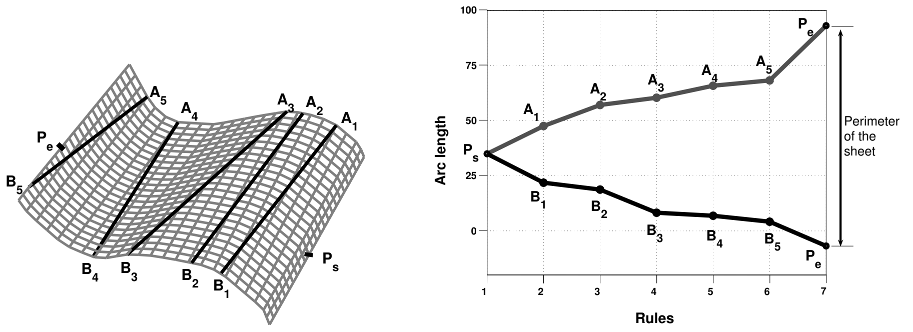


Figure 3: (left) The generated mesh with the control points (A_i, B_i). (right) Arc lengths s_A and s_B of the control points with the interpolating functions f_A and f_B.

Table 1 summarizes the model parameters. The model has $3+S+3n$ parameters, S being the number of parameters describing the mesh boundary (for instance, width and height in the case of a rectangular shape) and n being the number of guiding rules.

Parameters	**Description**	**Size**
n	number of guiding rules	1
n_e	number of extra rules	1
$\mathscr{S}$	mesh boundary parameters	S
P_s	arc length of the 'starting point'	1
P_e	arc length of the 'ending point'	1
s_A	arc lengths of the first point defining the guiding rules	n
s_B	arc lengths of the second point defining the guiding rules	n
θ	bending angles along the guiding rules	n

Table 1: Summary of the model parameters. (top) Discrete parameters (kept fixed during nonlinear refinement step). (bottom) Continuous parameters.

The deformation is parametrized by the guiding rules. Those are sorted from the 'starting point' to the 'ending point', making wavy the deformation. For example, it can not generate a sheet with the four corners pulled up. It is not however a significant drawbacks since the model is used for global object deformation recovery.

3 A Multiple View Fitting Algorithm

Our goal is to fit the model to multiple images. We assume that a 3D point set and camera pose have been reconstructed from image point features by some means. We use the reprojection error as an optimization criterion. As is usual for dealing with such a nonlinear criterion, we compute a suboptimal initialization that we iteratively refine.

3.1 Initialization

We begin by reconstructing a surface interpolating the given 3D points. A rule detection process is then used to infer our model parameters.

Step 1: Interpolating surface reconstruction. Details about how the 3D points are reconstructed are given in §4.1. The interpolating surface is represented by a 2D to 1D Thin-Plate Spline function [2], mapping some planar parameterization of the surface to point height. Defining a regular grid on the image thus allows us to infer the points on the 3D surface. Figure 4 and Figure 6 show two examples.

Step 2: Model initialization by rule detection. The model is initialized from the 3D surface. The side length is choosen as the size of the 3D mesh.

Guiding rules must be defined on the surface. This set of n rules must represent the surface as accurately as possible. In [3] an algorithm is proposed to find a rule on a given surface. We use it to compute rules along sites lying on the diagonal, the horizontal and the vertical axes. These sites are visible on Figure 4.

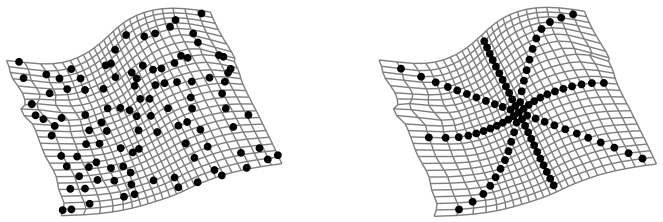


Figure 4: Model initialization. (left) Reconstructed 3D points and the interpolating surface. (right) Points where rules are sought.

The remaining rules are described by the arc length of their intersection points with the mesh boundary. The two arc lengths defining a rule i can be interpreted as a point R_i in $\mathbb{R}^2$, as shown in Figure 5 . Our goal is now to find the vectors s_A and s_B such that their interpolating functions f_A and f_B, defining the parametric curve (f_A, f_B) in $\mathbb{R}^2$, describe the rules. We thus compute s_A and s_B such that the distance between the curve (f_A, f_B) and the points R_i is minimized.

This gives the n guiding rules. The bending angle vector θ is obtain from the 3D surface by assuming it is planar between two consecutive rules. The initial suboptimal model we obtain is shown on Figure 6.

3.2 Refinement

The reprojection error describes how well the model fits the actual data, namely the image feature points. We thus introduce latent variables representing the position of each point

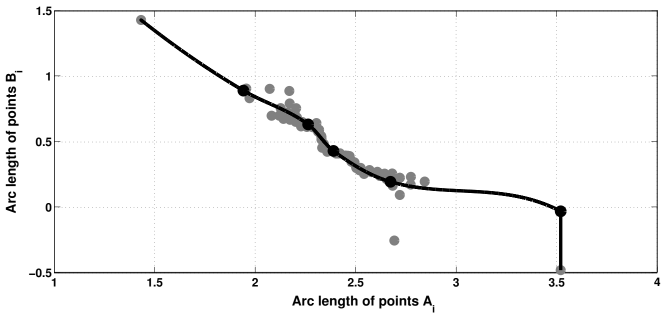


Figure 5: The points in gray represent the detected rules. The black curve is the parametric curve (f_A, f_B) and the black points are the estimated controls points that define the initial rules.

onto the modeled mesh with two parameters. Let L be the number of images and N the number of points, the reprojection error is:

$$e = \sum_{i=1}^{N} \sum_{j=1}^{L} (m_{j,i} - \Pi(C_j, M(S, x_i, y_i)))^2. \quad (2)$$

In this equation, $m_{j,i}$ is the i-th feature point in image j, $\Pi(C,M)$ projects the 3D point M in the camera C and $M(S, x_i, y_i)$ is a parameterization of the points on the surface, with S the surface parameters. The points on the surface are initialized by computing each (x_i, y_i) such that their individual reprojection error is minimized, using initial surface model.

To minimize the reprojection error, the following parameters are tuned: The parameters of the model (the number of guiding and extra rules is fixed), see Table 1, the pose of the model (rotation and translation of the generated surface) and the 3D point parameters.

The Levenberg-Marquardt algorithm [5] is used to minimize the reprojection error. Upon convergence, the solution is the Maximum Likelihood Estimate under the assumption of an additive *i.i.d.* Gaussian noise on the image feature points.

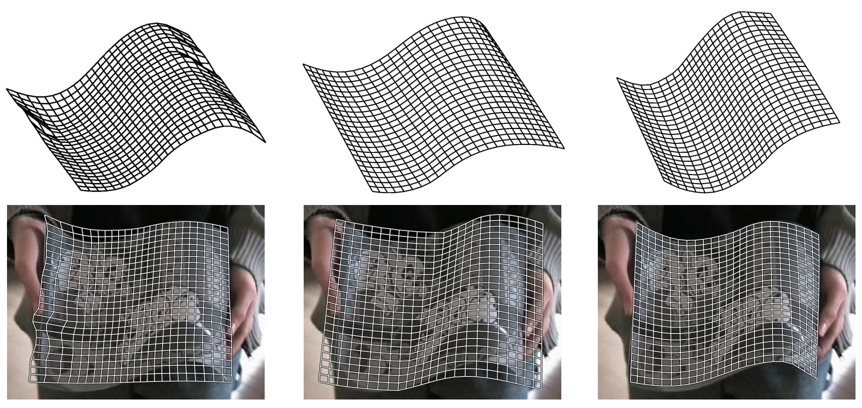


Figure 6: (top) 3D surfaces. (bottom) Reprojection into images. (left) Interpolated surface. (middle) Initialized model. (right) Refined model.

4 Experimental Results

We demonstrate the representational power of our fitting algorithm on several sets of images. First, we present the computation of a 3D point cloud. Second, we show the results for the three objects we modeled. Third, we propose some augmented reality illustrations.

4.1 3D Points Reconstruction

The 3D point cloud is generated by triangulating point correspondences between several views. These correspondences are obtained while recovering camera calibration and pose using Structure-from-Motion [5]. Points off the object of interest and outliers are removed by hand. Figure 4 shows an example of such a reconstruction.

4.2 Model Fitting

Even if our algorithm deals with several views, the following results have been performed with two views. Figure 6 and Figure 7 show the 3D surfaces, their reprojection into images and the reprojection errors distribution for the paper sequence after the three main steps of our algorithm: The reconstruction (left), the initialization (middle) and the refinement (right). Although the reconstruction has the lowest reprojection error, the associated surface is not satisfying, since it is not enough regular and does not fit the borders of the sheet. The initialization makes the model more regular, but is not enough accurate to fit the boundary of the paper, so that important reprojection errors remain. At last, the refined model is visually acceptable and its reprojection error is very close to the reconstructed one. It means that our model accurately fits the image points, while being governed by a much lower number of parameters than the set of independant 3D points has. Moreover the reprojection error significantly decreases thanks to the refinement step, which validates relevance of this step.

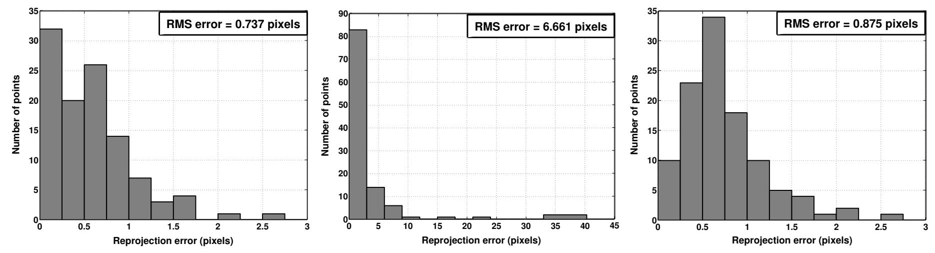


Figure 7: Reprojection errors distribution for the images shown in Figure 6. (left) 3D point cloud. (middle) Initial model. (right) Refined model.

We have tested our method on images of a poster. The results are shown in Figures 8. The reprojections of the computed model are acceptable: The reprojection error of the reconstruction is 0.35 pixels and the one for the refined model is 0.59 pixels.

At last, we fit the model to images of a rug. Such an object does not really satisfy the constraints of developable surfaces. Nevertheless, it is stiff enough to be well-approximated by our model. The results are thus slightly less accurate than for the paper

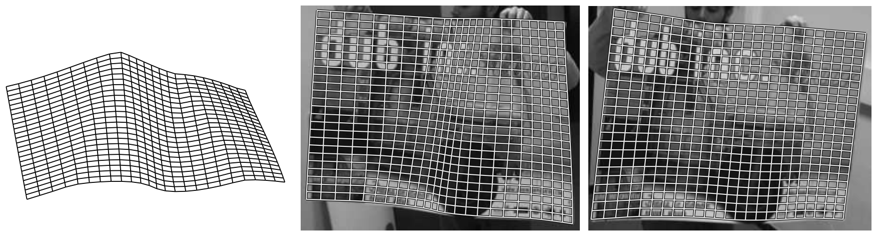


Figure 8: Poster mesh reconstruction. (left) Estimated Model. (middle) Reprojection onto the first image. (right) Reprojection onto the second image.

and the poster: The reprojection error of the reconstruction step is 0.34 pixels and the one of the final model is 1.36 pixels. Figure 9 shows the reprojection of the model onto the images used for the reconstruction.

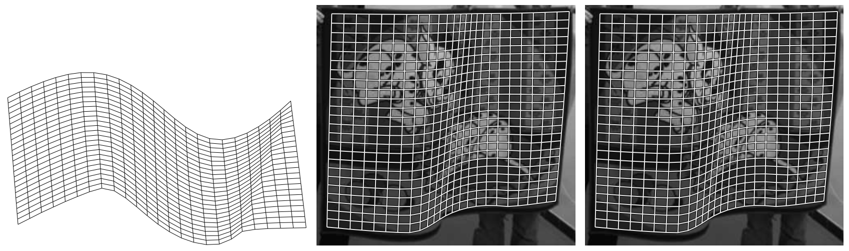


Figure 9: Rug mesh reconstruction. (left) Estimated Model. (middle) Reprojection onto the first image. (right) Reprojection onto the second image.

4.3 Applications

We demonstrate the proposed model and fitting algorithm by unwarping and augmenting images, as shown on Figures 10 and 11. Knowing where the paper is projected onto the images allows us to change the texture map or to overlay some pictures. The augmenting process is described in Table 2. Since we estimate the incoming lighting, the augmented images look realistic.

```
AUGMENTING IMAGES
    1. Run the proposed algorithm to fit the model to images
    2. Unwarp one of the images chosen as the reference one to get the texture map
    3. Augment the texture map
    4. For each image automatically do
        4.1 Estimate lighting change from the reference image
        4.2 Transfer the augmented texture map
```

Table 2: Overview of the augmenting process.

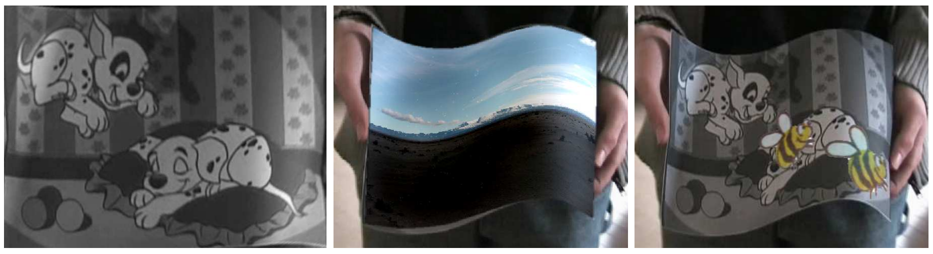


Figure 10: Some applications. (left) Unwarped texture map of the paper. (middle) Changing the whole texture map. (right) Augmented paper.

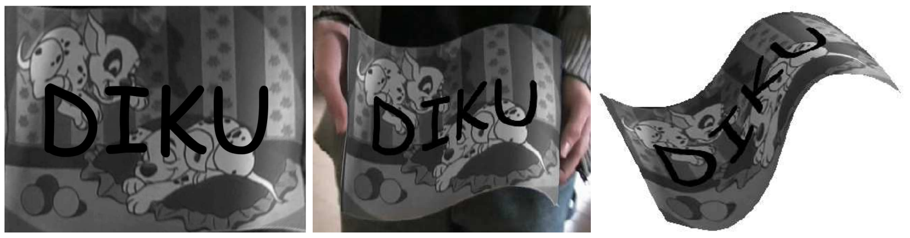


Figure 11: Augmentation. (left) Augmented unwarped texture map. (middle) Augmented texture map in the first image. (right) Synthetically generated view of the paper with the augmented texture map.

5 Conclusion and Future Work

This paper describes a quasi-minimal model for paper-like objects and its estimation from multiple images. Although there are few parameters, the generated surface is a good approximation of smoothly deformed paper-like objects. This is demonstrated on real image sequences thanks to a fitting algorithm which initializes the model first and then refines it in a bundle adjustment manner.

There are many possibilities for further research. The proposed model could be embedded in a monocular tracking framework or used to generate sample meshes for a learning-based model construction.

We currently work on alleviating the model limitations mentioned earlier, namely handling a general boundary shape and the comprehensive set of feasible deformation.

References

[1] V. Blanz and T. Vetter. Face recognition based on fitting a 3D morphable model. IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 25(9), September 2003.

[2] F. L. Bookstein. Principal warps: Thin-plate splines and the decomposition of deformations. IEEE *Transactions on Pattern Analysis and Machine Intelligence*, 11(6):567–585, June 1989.

[3] H.-Y. Chen and H. Pottmann. Approximation by ruled surfaces. *Journal of Computational and Applied Mathematics*, 102:143–156, 1999.

[4] N. A. Gumerov, A. Zandifar, R. Duraiswami, and L. S. Davis. Structure of applicable surfaces from single views. In *Proceedings of the European Conference on Computer Vision*, 2004.

[5] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003. Second Edition.

[6] Y. Kergosien, H. Gotoda, and T. Kunii. Bending and creasing virtual paper. IEEE *Computer Graphics & Applications*, 14(1):40–48, 1994.

[7] S. Leopoldseder and H. Pottmann. Approximation of developable surfaces with cone spline surfaces. *Computer-Aided Design*, 30:571–582, 1998.

[8] M. Pilu. Undoing page curl distortion using applicable surfaces. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, December 2001.

[9] H. Pottmann and J. Wallner. Approximation algorithms for developable surfaces. *Computer Aided Geometric Design*, 16:539–556, 1999.

[10] M. Salzmann, S. Ilic, and P. Fua. Physically valid shape parameterization for monocular 3-D deformable surface tracking. In *Proceedings of the British Machine Vision Conference*, 2005.

[11] M. Sun and E. Fiume. A technique for constructing developable surfaces. In *Proceedings of Graphics Interface*, pages 176–185, May 1996.

A Comparison of Techniques for Approximating Full Image-Based Lighting

Claus B. Madsen and Rune E. Laursen
Laboratory of Computer Vision and Media Technology
Aalborg University, Aalborg, Denmark
cbm@cvmt.aau.dk
www.cospe.dk

Abstract

Image-Based Lighting (IBL) has become a very popular approach in computer graphics, especially for special effects, such as insertion of virtual objects into real imagery (Augmented Reality). In essence IBL is based on capturing the illumination conditions in a scene in an omnidirectional image, such that the image describes intending radiance at a point from all directions. Such an omnidirectional measurement of radiance is typically called a light probe image. Using the illumination information from such an image virtual objects can be rendered with consistent shading including global illumination effects such as color bleeding.

Rendering with light probe illumination is extremely time consuming. Therefore a range of techniques exist for approximating the intending radiance described in a light probe image by a finite number of directional light sources. We describe 4 such techniques from the literature and perform a comparative evaluation of them in terms of how well they each approximate the final irradiance as a function of how many sources they are allowed to use in the approximation. We demonstrate that for relatively low numbers of sources (e.g., less than 100 sources) one particular method performs significantly better than the three other techniques.

1 Introduction

Image-based approaches have gained widespread popularity in computer graphics because of the inherent problems with purely model-based approaches, [11]. Image-based techniques have been used for 3D modeling of real scenes (Image-Based Modeling), for rendering from a bank of images with no 3D model whatsoever (Image-Based Rendering), and for modeling the complex illumination conditions in real scenes (Image-Based Lighting). The latter has especially been applied for special effects, i.e, rendering virtual objects into real imagery for movies, commercials, etc.

Image-Based Lighting (IBL) has become an extremely frequently used technique since it in an intuitively simple manner allows you to render a virtual object with illumination conditions that perfectly match those of a real scene. The idea is simply that you use a camera to measure the light arriving at some point in the scene, that point it which you want to insert a virtual object. In practice people most often use a polished steel ball, place it somewhere in a scene, and take an image of it with a tele-lens from some distance away. After cropping away everything which is not a projection of a point on the sphere the image now contains information about how much light arrives at the position of the ball from all possible directions. In other words the ball image contains information about the incident radiance (field radiance) at the ball location. Figure 1 illustrates this concept.

Using reflective sphere for acquisition of light probes is the standard approach and employed by most researchers in the field for its simplicity. Most of the light probes shown in this paper have been downloaded from Debevec's probe gallery, [2], and have been acquired by merging two views of a reflective sphere in order to avoid the reflections of the camera and the photographer in the final light probe, and in order to avoid the problem with a small ”blind region” behind the sphere.

When we acquire our own light probes we use Sigma 180 degree field-of-view fish eye lens. By taking two such images in opposing direction we can merge them together to a complete spherical image using the HDRShop program, [7]. With this approach we get much higher resolution light probes, and avoid the smearing of detail that an imperfect mirror ball can result in. Figure 2 shows an example of two such hemi-spherical images.

Regardless of whether the light probe is acquired with a reflective sphere or with multiple views with a fish-eye lens it is still very important to handle the dynamic range of the light in the scene. This problem is handled by acquiring the same view at multiple different exposures, gradually lowering the exposure time until no pixels in the image are saturated. These multiple exposure are then fused into a single High Dynamic Range (HDR) floating point image, [6].

Once a light probe has been acquired at some position in some scene it can be used for many purposes. The light probe is a map of the incident radiance at the acquisition

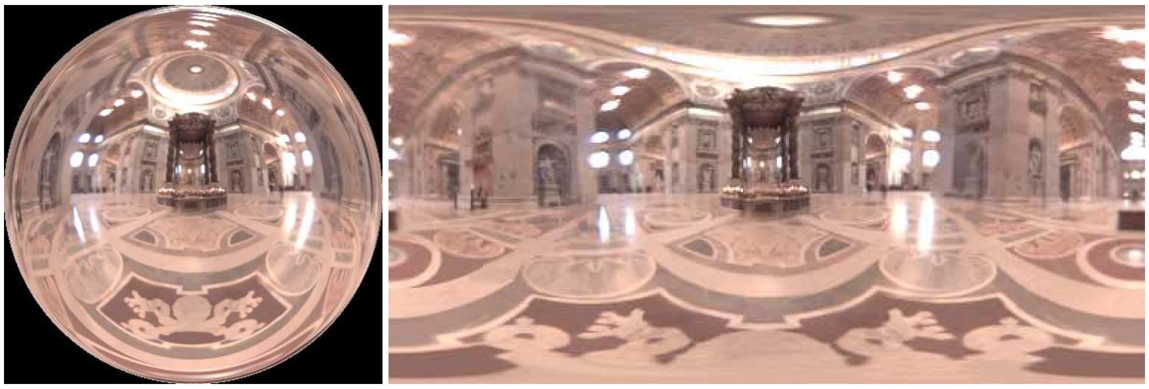


Figure 1: Left: the light probe image is based on a cropped image of a reflective sphere. Right: light probe images are omni-directional, i.e., cover the entire sphere around the light probe position. Here we have remapped the light probe image to longitude-latitude format, where the a full 360 degrees are represented along the horizontal (longitude) axis, and 180 degree are represented along the vertical (latitude) axis. For construction and remapping of light probe images we use HDRShop 2.09, [7].

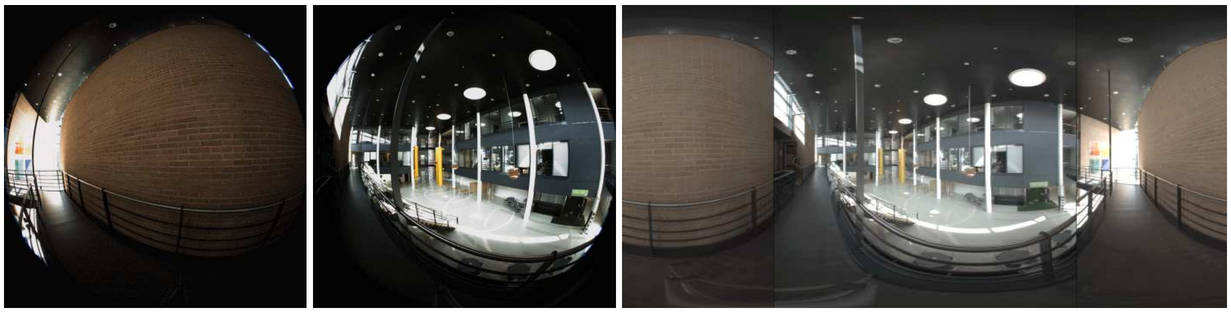


Figure 2: Left and center: two semi-spherical images acquired with a Sigma 180 degree field-of-view fish-eye lens in opposing directions. Right: the two semi-spherical images merged and mapped as a longitude-latitude light probe using HDRShop 2.09, [7].

point. Each pixel in the map corresponds to a certain direction and solid angle, and together all pixels cover the entire sphere around the acquisition point. A light probe can thus also be called a radiance map, or an environment map. With this information virtual objects can be rendered into the scene with scenario consistent illumination e.g., [3, 4, 8]. Light probes can also be used to estimate the reflectance distribution functions of surfaces from images, as demonstrated in [13, 12]. For a review of illumination models in mixed reality see [9].

Actually using light probes for rendering is computationally very heavy. For a full global illumination rendering with path tracing using image-based lighting is extremely time consuming in order to reduce the noise level in the final rendering, simply because the light probe has to be treated as a spherical area light source enclosing the entire scene. To get a noise free estimate of the irradiance at a certain point requires thousands and thousands of samples of this area source.

To combat this problem several approaches have been proposed which take a light probe a attempts to approximate its illumination by a relatively low number of directional light sources. That is, the idea of these approaches is to find directions and the radiances of some number, say 64, directional light sources, such that the combined illumination from these sources approximate the combined illumination from the entire light probe.

With such a directional light source approximation to a light probe, Image-Based Lighting using light probes can also be implemented in real-time applications taking into account that each source causes shadows to be cast.

The aim of the present paper is simply to test the performance of these approximation techniques in terms of how well they actually approximate the light probe for a given number of sources.

The paper is organized as follows. First section 2 gives a very brief overview of the approach and results in the paper. In section 4 we give a brief description of four different approaches to using N directional light sources to approximate the illumination modeled by a light probe image. Section 5 then tests these four techniques in terms of their relationships between approximation error and number of sources used. Conclusions and directions for future research are given in section 6.

2 Overview of the idea of this work

Figure 3 shows a light probe together with the result from one of the approximation techniques we study in this paper. In this particular case the technique has been allocated 8 directional sources which it has then distributed across the light probe longitude-latitude map in an attempt to capture the radiance distribution of the original light probe. Naturally, the accuracy of the approximation depends on the number of sources allocated. The original light probe is simply xres times yres directional sources, where xres is the number of pixel in the longitude direction, and yres is the number of pixels in the latitude direction.

We then run any given technique on some light probe image to produce approximations with 2, 4, 8, 16, etc. light sources. Given these sets of approximated sources we compute what the resulting error in irradiance is compared to ground truth, which in this case is the irradiance computed by using the radiance from all pixels in the light probe.

Before we proceed with the actual techniques and there performances we need to establish a small theoretical basis.

3 Terminology

In more formal terms we can say that the light probe image is a spatially discrete measurement of the continuous function describing the incident radiance (measured in $W/(m^2 \cdot Sr)$, which in turn is a function of the incident direction. Using standard spherical coordinates a direction in space is represented by two angles, θ and ϕ, where θ is the angle the direction vector makes with the coordinate system z-axis (latitude), and ϕ is the angle the projection of the vector on the xy-plane makes with the x-axis. Therefore the incident radiance can be written as:

$$L(\theta, \phi) \tag{1}$$

In this paper we will exclusive use the latitude-longitude mapping (LL mapping) of light probe images. To establish a relationship between spherical angles and points in the LL map we will define that the middle row in the image corresponds to the equator of the unit sphere, i.e, corresponds to $\theta = \pi/2$, the top row corresponds to $\theta = 0$ and the bottom row corresponds to $\theta = \pi$. Moreover we can arbitrarily define $\phi = 0$ to correspond the middle column in the LL map with negative ϕ values right of that column.

Consider a differential area at the position of the light probe acquisition with the z-axis as surface normal. The irradiance, E, on that surface is given by:

$$E = \int_0^{2\pi} \int_0^{\pi/2} L(\theta, \phi) \cos(\theta) \sin(\theta) d\theta d\phi \tag{2}$$

In general, let (θ_k, ϕ_k) denote the orientation of an arbitrary normal, and let Ω_k denote the semi-sphere defined by that normal, (a semi-sphere with top point in the normal direction). Correspondingly, $E(\theta_k, \phi_k)$ shall denote the irradiance on a differential surface with such a normal.

Returning again to LL maps, the LL mapping is a non-uniform sampling of the sphere, since it is severely over-sampled at the poles. To correct for that we must multiply the RGB values of all light probe pixels in the LL map with $\sin\theta$. After this we can treat all pixels with equal weight, e.g., by transferring the radiance of one pixel to the location of another by sampling adding its value to the other pixel, as most of the approximation techniques described below do. If the image acquisition process is photometrically calibrated each pixel will actually represent a physically correct radiance in $W/(m^2 \cdot Sr)$, but we have mapped this measurement to an LL map in some resolution (longitude resolution, l, times latitude resolution, m), so we have to take into account what solid angle an LL map pixel corresponds to. The LL map is a rectangular map that covers 2π radians in longitude, and π radians in latitude, so the total solid angle is $2\pi^2$. This area is divided evenly between the l times m pixels, so by dividing every LL map pixel with $2\pi^2/(l \cdot m)$ ensures that we can sum up pixels easily when we wish to perform radiance integration for irradiance computations.

For example, the irradiance for a differential area surface with a normal given by (θ_k, ϕ_k) can be calculated as follows, if we use x and y to denote pixel integer coordinates, and denote the light probe image pixel values by $P(x, y)$:

$$E(\theta_k, \phi_k) \approx \sum_{(u,v) \in \Omega_k} P(u, v) \cos(\psi) \tag{3}$$

where ψ is the angle between the surface normal and the direction vector corresponding to (x, y), and where we use $\approx$ to stress the fact that here we are dealing with an approximation to a surface integral based on a sum of discrete samples.

The techniques we are evaluating all produce a set of N directional light sources, where the ith source has radiance L_i, and direction vector given by given by $(\theta_i, \phi_i$. Similarly to Eq. 3 the incident irradiance from such a set of sources can be calculated as:

$$E(\theta_k, \phi_k) = \sum_{i=1}^{N} L_i \cdot max(0, \cos(\psi)) \tag{4}$$

where ψ again is the angle between the surface normal and the direction vector to the light source, and where we

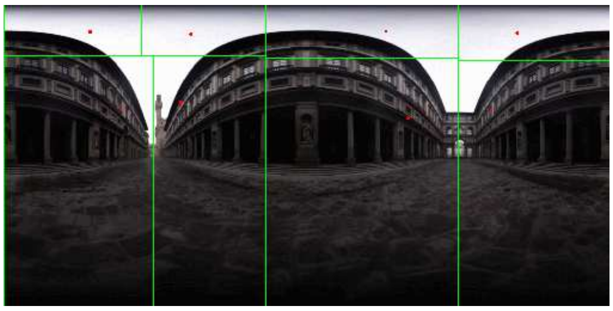


Figure 3: Result from running the Median Cut approximation technique using 8 directional sources on a light probe. Each rectangular region contains a red dot. This red dot marks the chosen direction for a particular directional source, and all the combined radiance from the region has been transferred to this particular source direction.

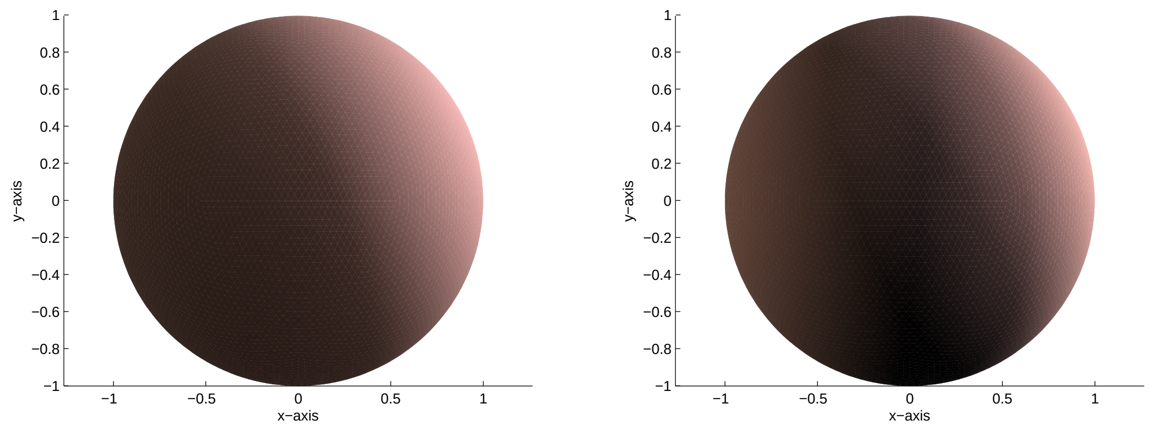


Figure 4: Left: ground truth irradiance for around 20000 normals distributed evenly on a sphere computed for the Galileo's Tomb light probe. Right: irradiances resulting from running the Median Cut source approximation technique to produce 8 directional sources. On print the difference may be visually subtle, but the average error is actually around 25 percent, and the maximum error is more than 75 percent.

have clamped $\cos(\psi)$ to non-negative values to rule out negative irradiance contributions from sources outside the semi-sphere defined by the normal in question.

4 Light probe approximation techniques

As mentioned previously there exist a number for approaches to finding a set of directional light sources which approximate a full radiance map in the form of a light probe. We have found four different techniques, three of which are closely related and operate directly in the radiance space image domain of the light probe, in particular on the longitude-latitude mapping. The last technique is quite different in that it operates in *irradiance* space.

Below we briefly describe the four different techniques, starting with the radiance space techniques.

4.1 Lightgen

A description of this technique will be included in a future version of this paper. The normal citation given for the approach is [1].

4.2 Median cut

The median cut technique, [5], is conceptually wonderfully simple. The idea is to recursively split the LL map into regions of approximately equal summed radiance. Since the method splits all regions K times the techniques produces 2^K sources, i.e., 2, 4, 8, 16, 32, etc. Figure 3 illustrated the result of running the technique on a light probe. The algorithm is as follows:

1. Add the entire light probe image to the region list as a single region.

2. For each region in the list, subdivide along the longest dimension such that its light energy is divided evenly.

3. If the number of iterations is less than K, return to step 2.

4. Place a light source at the centroid of each region, and set the light source radiance to the sum of the pixel values within the region.

The strength of this approach is that it is so straight forward, computationally light and easy to implement. The problems with this approach lies in two issues. The first issue is that it subdivides all regions at each iteration and depending on K there can be a large jump in the number of sources, which may be disadvantageous for real-time rendering with the approximated sources, where one would like as many sources as possible, but at the same time there is a performance limit in the graphics hardware. The second issue relates to step 4, where, for small K, and thereby large regions, a lot of radiance is moved quite large distances over the sphere surface.

4.3 Adaptive median cut

A description of this technique will be included in a future version of this paper. We have developed this technique which is heavily based on the original Median Cut technique, but our version can produce any number of sources, not just a power of 2.

4.4 Irradiance Optimization

The Irradiance Optimization technique by Madsen et al., [10], is significantly different from the first three. While the first three all operate entirely on a pixel level in the light probe image, i.e., operate in radiance space, the Madsen method operates in *irradiance* space.

The method is based on first using the original light probe image to compute the ground truth irradiance for a large number (M) of normal directions uniformly distributed across the unit sphere using Eq. 3. These M irradiance values constitute the goal vector in an optimization to estimate the parameters of N directional sources. Each source is defined by five parameters (RGB radiances and two direction angles).

Given an estimate of these 5 times N parameters it is possible to compute the approximated irradiances for the M normals using Eq. 4. By comparing the approximated radiances to the ground truth radiances we obtain an error vector, which can be converted to a parameter update vector. The source estimation process is this an iterative, non-linear optimization process based on Newton's iterative method, since the Jacobian can be expressed analytically.

5 Comparative evaluation

In a future version of this paper we will have compiled more extensive evaluations. At present we have only tested two methods (Median Cut and Irradiance Optimization), and they have just been evaluated on one light probe image. In this section we describe the results from these initial evaluations, and offer some observations based on them.

5.1 Tested light probes

The evaluation documented in this paper is based on the light probe shown in figure 5.

5.2 Performances

We ran the Median Cut and the Irradiance Optimization methods on the test light probe, and produced directional source approximations with 2, 4, 8, 16, 32, 64, and 128 sources. The Irradiance Optimization technique can be produce any number of sources, but was constrained to the source number cases which where also feasible for the Median Cut approach. We were unable to obtain a convergence on a 128 source solution with the Irradiance Optimization technique. This will be discussed later.

The evaluation is based on computing the irradiances resulting from the estimated set of sources for a large number of surface normal evenly distributed on a unit sphere, and comparing them to the ground truth irradiances. For each color channel when then compute the mean and the maximum of the absolute differences between estimated and ground truth irradiances. Figure 6 shows curves representing average and maximum error for each of the two methods as a function of the number of light sources used. The errors are an average over RBG.

5.3 Discussion

Figure 6 clearly shows that the Irradiance Optimization techniques performs much better than the Median Cut method. Generally the Median Cut method requires 2 to 3 times as many sources to achieve the same error as the Irradiance Optimization technique. For rendering this is very important from a computational point of view, since it will always be an advantage to use as few sources as possible.

In this test it was seen that the Irradiance Optimization technique could not converge when the number of sources comes above some threshold (64). This is a general tendency we have noticed, and it is strongly believed to be related to the fact that when number of sources grows too high there is too little energy (irradiance) for some sources to latch on to. Very quickly in the iterations the dominant sources become stable, leaving ever smaller amounts

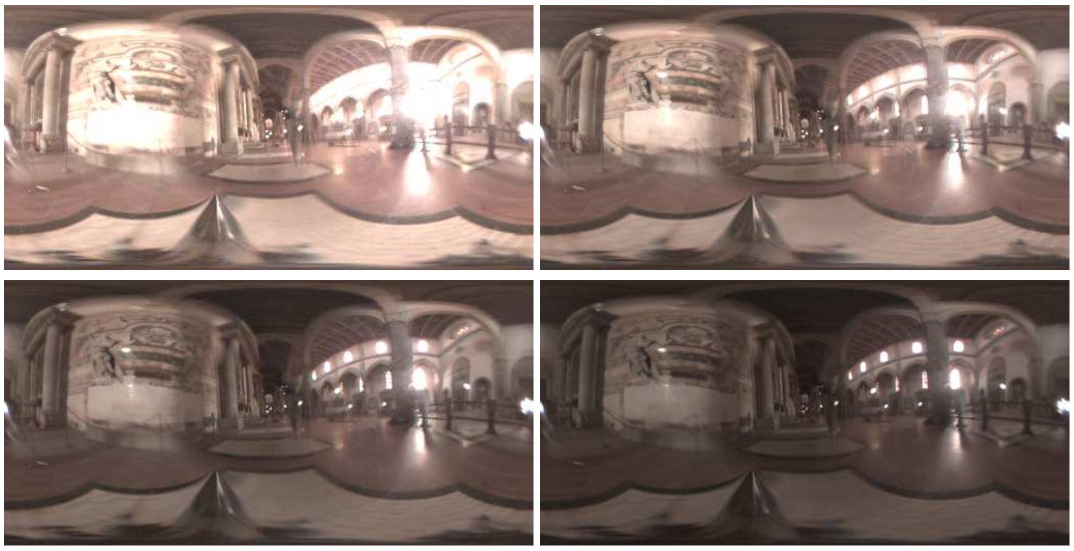


Figure 5: The tested light probe: Galileo's tomb in Florence, Italy. Acquired from [2]. Here shown in four different exposures to illustrate dynamic range.

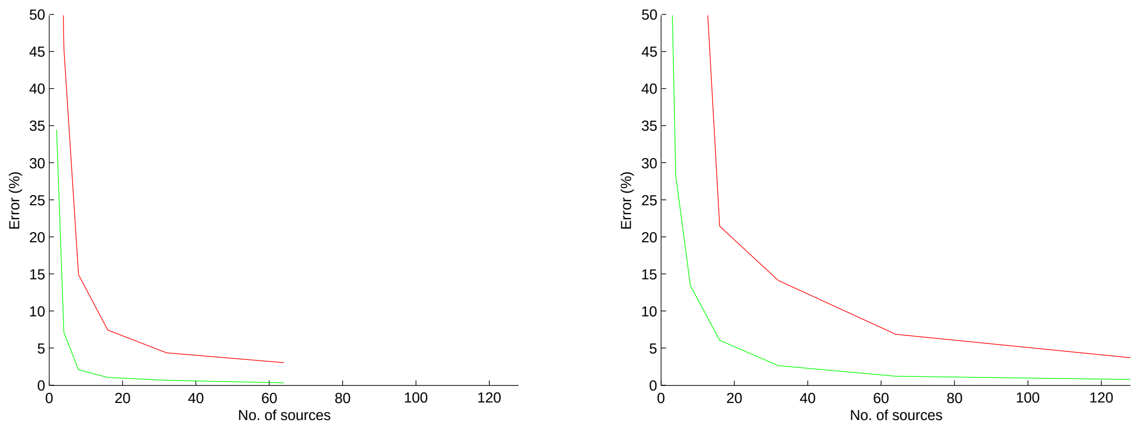


Figure 6: Mean and max irradiance error in percent for the two tested techniques. Left: Irradiance Optimization. Right: Median Cut.

of energy to distribute among the rest of the sources being estimated. At the same time the sources tend to repel each other when they distribute across the sphere, because each source has a semi-spherical "'footprint"' in the irradiances, so each source is like shining a torch on a sphere, and sources will be reluctant to overlap footprints too much.

6 Conclusions

We have demonstrated that there is significant differences in the performance of the available techniques to approximate light probes radiance maps with a set of directional light sources. Test so far clearly demonstrate that the Irradiance Optimization technique requires much less sources to achieve the error level as the other techniques.

Future work includes several straight forward issues, plus one somewhat more complicated. Primarily we need to test all the techniques, and we need to do it on more qualitatively different light probes. In this regard we are thinking about both indoor and outdoor scenarios. So far we have only tested the techniques in terms of resulting irradiance. Future experiments will attempt to test not only irradiance but also the spatial distribution of incident radiance. We plan to evaluate this issue by rendering scenes with the approximated sources and test the reflected radiance in and around cast shadows and compare them to shadows rendered with Monte Carlo path tracing.

Acknowledgments

This research is funded by the CoSPE project (26-04-0171) under the Danish Research Agency. This support is gratefully acknowledged.

References

[1] J. M. Cohen and P. Debevec. *The LightGen HDRShop plugin*, 2001. www.hdrshop.com/main-pages/plugins.html.

[2] P. Debevec. *www.debevec.org/probes*.

[3] P. Debevec. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *Proceedings: SIGGRAPH 1998, Orlando, Florida, USA*, July 1998.

[4] P. Debevec. Tutorial: Image-based lighting. *IEEE Computer Graphics and Applications*, pages 26 – 34, March/April 2002.

[5] P. Debevec. A median cut algorithm for light probe sampling. In *Proceedings: SIGGRAPH 2005, Los Angeles, California, USA*, August 2005. Poster abstract.

[6] P. Debevec and J. Malik. Recovering high dynamic range radiance maps from photographs. In *Proceedings: SIGGRAPH 1997, Los Angeles, CA, USA*, August 1997.

[7] P. Debevec et al. *www.hdrshop.com*.

[8] S. Gibson, J. Cook, T. Howard, and R. Hubbold. Rapic shadow generation in real-world lighting environments. In *Proceedings: EuroGraphics Symposium on Rendering, Leuwen, Belgium*, June 2003.

[9] K. Jacobs and C. Loscos. State of the art report on classification of illumination methods for mixed reality. In *EUROGRAPHICS*, Grenoble, France, September 2004.

[10] C. B. Madsen, M. K. D. Sørensen, and M. Vittrup. Estimating positions and radiances of a small number of light sources for real-time image-based lighting. In *Proceedings: Annual Conference of the European Association for Computer Graphics, EUROGRAPHICS 2003, Granada, Spain*, pages 37 – 44, September 2003.

[11] M. M. Oliveira. Image-based modelling and rendering: A survey. *RITA - Revista de Informatica Teorica a Aplicada*, 9(2):37 – 66, October 2002. Brasillian journal, but paper is in English.

[12] Y. Yu, P. Debevec, J. Malik, and T. Hawkins. Inverse global illumination: Recovering reflectance models of real scenes from photographs. In *Proceedings: SIGGRAPH 1999, Los Angeles, California, USA*, pages 215 – 224, August 1999.

[13] Y. Yu and J. Malik. Recovering photometric properties of architectural scenes from photographs. In *Proceedings: SIGGRAPH 1998, Orlando, Florida, USA*, pages 207 – 217, July 1998.

Shadow Segmentation and Augmentation Using α-overlay Models that Account for Penumbra

Michael Nielsen and Claus B Madsen

Aalborg University, Computer Vision and Media Technology Laboratory, Niels Jernes Vej 14, DK-9220 Aalborg, Denmark

Abstract

This paper introduces a new concept within shadow segmentation. Previously, an image is considered to consist of shadow and non-shadow regions. Thus, a binary mask is estimated using various heuristics regarding structural and retinex/color constancy theories. We wish to model natural shadows so that an augmented virtual object can cast an exact shadow. The penumbras (half-shadows) must be taken into account so that we can model the soft shadows. We hope to achieve this by modelling the shadow regions (umbra and penumbra alike) with a transparent overlay. This paper reviews the state-of-the-art shadow theories and presents two overlay models. These are analyzed analytically in relation to color theory and tangibility.

Key words: Computer Vision, Shadows

1 Introduction

The methods that are investigated in this paper are part of an idea to augment real images with virtual objects. If these have to look believable their shadows must look like the real shadows. For this purpose the light sources must be known and they can be estimated by detecting the real shadows.

We want to find a model that can be used for shadow segmentation as well as shadow synthesis. If the model is a good model of the physical shadows, we can assume that the real shadows were generated by the same model and a detection of such shadows would be an estimation problem of the parameters in the shadow model. Thus, the fewer parameters to estimate in the model the better.

1.1 State of the Art

There is some work done in the field from 2000 through 2006. The density is concentrated around 2005-2006.

We can classify the solutions into single image based vs. multiple image based and natural (un-augmented) scenes vs. augmented scenes. Furthermore, the degree of known information about the scene and the need for heavy calibration procedures have to be taken into account.

Our aim is to make a purely pixel driven method which works on single images in un-augmented scenes with no knowledge about the scene. Usually outdoor scenes with blue skylight and direct sun is assumed. Even though the aim is unsupervised segmentation we will consider some simple kind of user interaction such as (Wang and Cohen, 2005). They used a few strokes from the user to train gaussian mixture models in order to segment complex foreground objects with alpha channels such as hair and spider webs using graph cuts.

(Barnard and Finlayson, 2000) uses chromaticity color segmentation and investigates the edges between the segments. The color ratio is used with other tests to determine the probability that the edge is an illumination boundary or a material boundary. The probability is given by how well the color ratio jump matches a cone in 3D space generated from 100 cases of typical illumination and whether the jump is seen between other regions as well. It also assumes soft gradient boundaries as being shadow boundaries. The reconstruction of the shadowless scene had artifacts.

(Salvador et al., 2001, 2004) distinguished between cast shadows (onto the ground plane) and self shadow. The detection relied on the edge image of a c_1, c_2, c_3 chromaticity image defined by equation 1. It is similar to normalized rgb space, but it is linear confined to $[0 - \frac{\pi}{2}[$.

$$c_i = \arctan(\frac{c_i}{\max(c_k, c_l)}) \quad \text{where } i \neq k \neq l \tag{1}$$

They considered dark pixels a-priori to be shadows and corrected this belief using heuristics concerning the edges of the real image and edges of the c_i image. This worked well in images with controlled simple geometry. It was tested with still images (of fruit) and video (moving people and cars). Considering the simplicity of this approach and the fact that it requires no assumptions about the camera, the results are impressive.

(Madsen, 2003) described shadows as an RGB alpha overlay, see equation 2. It is not just a black layer with an alpha channel, because the shadows are not only darker versions of the illuminated areas, but there is a change of hue, caused by the difference in hue between direct and ambient light. There is a fixed alpha for any given region. RGB_r is the resulting image with shadow, RGB_o is the original image, and RGB_a is an overlay. i labels an individual pixel. α can be described as the degree of shadow and the overlay color relates to the the tonal and intensity change of the shadow.

$$\begin{bmatrix} R_i \\ G_i \\ B_i \end{bmatrix}_r = \alpha \begin{bmatrix} R_i \\ G_i \\ B_i \end{bmatrix}_a + (1-\alpha) \begin{bmatrix} R_i \\ G_i \\ B_i \end{bmatrix}_o \tag{2}$$

Furthermore, shadows are characterized as full shadow, *umbra*, and half shadow *penumbra*, assuming only one light source. Multiple light sources would generate more complex grades of shadow regions (more variation in an alpha layer and RGB_a).

Another direction takes advantage of planckian light and retinex theory. Assuming a single direct light source and another ambient light (different hue) (Finlayson et al., 2002a)(Finlayson et al., 2002b) computes a 1-d invariant image from the known path (or offset or retinex path) the shadow imposes on a log-ratio chromaticity plot. Note that ambient occlusion and surface normal direction is not taken into account in this model.

$$\hat{r}_k = log(\frac{\rho_k}{\rho_p}) \quad k \neq p \tag{3}$$

The known path/offset is to be pre-calibrated. The edge maps of this invariant image can be used just like in (Salvador et al., 2001). The recovery of a shadow free color image was worse than (Lu and Drew, 2005).

(Lu and Drew, 2005) continued Finlayson's work using graph cuts for optimizing the shadow mask. Their method finds a binary shadow mask and use the illumination invariant chromaticity transform (Finlayson et al., 2002a) as a static clue for computation of the capacities of the capacities in the graph model. They do not use any data term, but considers the brightness change as well as the chromaticity "shift" caused by the illumination color. It is not tested for difficult scenes and it does require knowledge of log illumination direction.

To explain this log illumination direction Finlayson model the camera response by equation 4. Sensor sensitivity q_k is assumed to be a dirac delta function. White-balance is considered part of q_k. The illumination is described by color temperature and Wien's approximation 5. The log chromaticity bands are given by 8. It assumes that γ actually is a pure gamma correction.

$$\rho_k = \gamma(E(\lambda_k)S(\lambda_k)q_k), \quad k = R, G, B \tag{4}$$

$$E(\lambda, T) \cong Ic_1\lambda^{-5}e^{\frac{c_2}{T\lambda}}, \quad \begin{matrix} c_1 = 3.74183E^{-16}mK \\ c_2 = 1.4388E^{-2}mK \end{matrix} \tag{5}$$

$$\rho_k = \gamma(Ic_1\lambda_k^{-5}e^{\frac{c_2}{T\lambda_k}}S(\lambda_k)q_k), \quad k = 1...3 \tag{6}$$

$$r_k = \rho_k/\rho_p \tag{7}$$

$$\acute{r}_k = log(r_k) = \gamma log(s_k/s_p) + \gamma(e_k - e_p)/T, \quad p = 1...3, k \neq p \tag{8}$$

$$s_k = Ic_1\lambda_k^{-5}S(\lambda_k)q_k \tag{9}$$

$$e_k = -c_2/\lambda_k \tag{10}$$

Equation 8 shows that in log chromaticity space the perception of color on a surface changes by altering the color temperature T. In a 2-d plot this forms a straight line, and the direction $\gamma(e_k - e_p)$. The model contains some major assumptions that might not hold any given camera:

- Narrow-band/Delta-function sensitivity (however, spectral sharpening may be insignificant (Barnard and Funt, 1998))
- Linearity (Note that gamma correction does not change the log illumination direction)
- The log illumination direction $e_k - e_p$ must be known (through calibration)
- Variation of the angle of surface and the visibility of hemisphere (ambient occlusion) is not taken into account.

It follows from this generalization that the color of a surface in full shadow is assumed to be a product its color in sunlight and a fixed shading factor (Finlayson et al., 2002b):

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix}_{shad} = \begin{bmatrix} \alpha R \\ \beta G \\ \gamma B \end{bmatrix}_{sun} \tag{11}$$

This relation holds for all pixels in the image. $[\alpha\beta\gamma]$ in this model relates to the RGB_a α-layer model in Eq.2, which is more versatile because it can weight the shading effect by *alpha* in penumbra areas and to take variation of the angle of surface and the visibility of hemisphere into account. See figure 1.

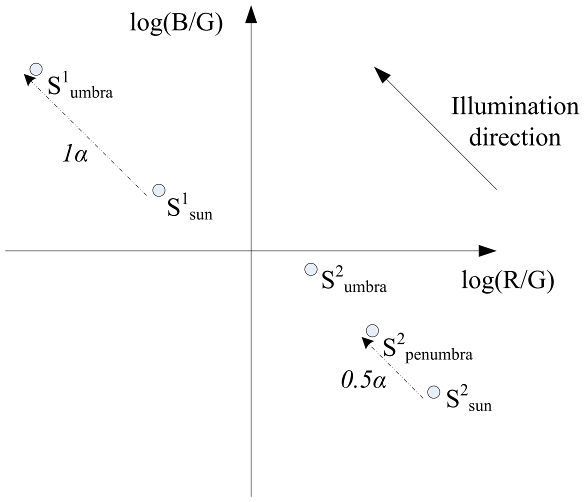


Fig. 1. 2-d log chromaticity space. The illumination direction in log chromaticity space shows the direction that a surface color moves in the 2-d plot from the color temperature changes from full sun to full shadow. It follows a straight line. Surface 1 (S^1) is plotted in sun and shadow (umbra). This relates to the alpha model as full α. We extend the model to account for varying degrees of shadow (penumbra and other factors). Surface 2 (S^2) is plotted in sun, half shadow, and full shadow. However, tonal changes from $\epsilon \rightarrow 0$ and inter-reflections do not map into the straight line. $\epsilon \rightarrow 0$ maps towards $[0, 0]$, while inter-reflections maps toward the colors of the reflecting surfaces.

1.2 Limitations

Consider the physics based model of camera response ρ in channel k to wavelengths λ in equation 12. The surface albedo $S(\lambda)$ is illuminated by the weighted irradiance E from the sky E_{sky} and the sun E_{sun}. Note that the E_{sky} is the integration of light from the entire hemisphere except where the sun is and ϵ accounts for occlusion of the hemisphere (ambient occlusion). This means that it is not valid in the case of a sunset, where the hemisphere can be bright and orange in the western half, and dark blue in the eastern half. In order to make the angular $(cos\theta)$ dependence of the sun explicit $\hat{E}_{sun}$ is a specific irradiance of the sun onto a perpendicular surface.

The camera sensors have certain sensitivities Q to the wavelengths λ. Then the camera applies a white balance correction W and a nonlinear dynamic range compression γ, e.g. a gamma 2.2 correction.

$$\rho_k = \gamma\left(w_k \int Q_k(\lambda)S(\lambda)(\epsilon E_{sky}(\lambda) + cos\theta \hat{E}_{sun}(\lambda))d\lambda\right), \quad k = R, G, B \tag{12}$$

It relates to the α-layer model very well, where α and $(1-\alpha)$ is a simplification of ϵ and $cos\theta$. *alpha* maps to the degree of shadow, and RGB_a maps to the tonal change that occurs because of (in priority):

(1) Hemisphere-sun color difference
(2) $\epsilon \to 0$
(3) Inter-reflections

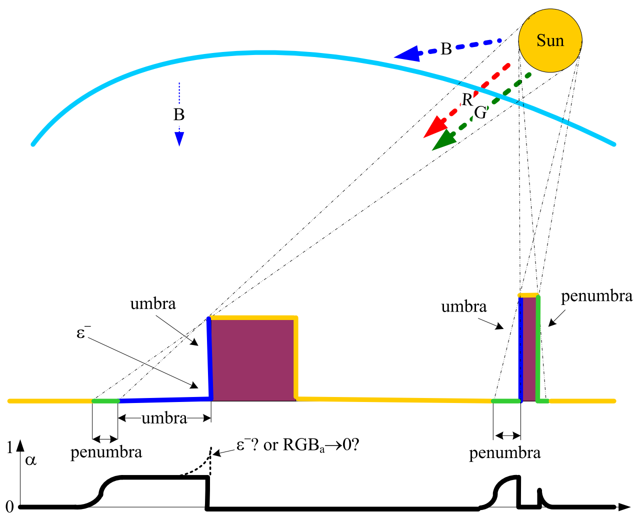


Fig. 2. This figure shows how umbra and penumbra regions can occur and how α should respond to those regions (in a photo taken directly from above). In the corner at the first box is a situation where the shadow becomes darker because the hemisphere is less accessible from those locations. The question remains whether this effect ($\epsilon \to 0$) should be modeled as a higher α or by adjusting the alpha overlay layer ($RGB_a \to 0$).

It would be interesting to see if a general (an average) transform for standard consumer cameras can be found which optimizes a *best guess* of the log illumination direction. But first we would like to assume that we know the camera sensitivities.

The sensors in the cameras are somewhat linear, but the human perception sensitivity is non linear. Cameras apply post processing to the linear raw data. sRGB and Adobe RGB formats have applied a gamma 2.2 function (which does not affect the direction of the log chromaticity space). However, cameras normally compress the dynamic range in an even more complicated manner, using e.g. an S-curve. In practice, there are some non-linearities in the darkest shadows and brightest highlights.

2 Materials and Methods

2.1 Shadow Model

Two models will be presented for investigation. Each will be analyzed analytically regarding the following questions:

- Shadow region as a function of the sunlit region?
- Sunlit region as a function of the shadow region?
- Considerations regarding the overlay color as a function of known parameters and extreme α's?

2.1.1 The additive model

Assuming equation 2 is a good model of applying shadows into the scene, the real shadows might as well have been applied the same way. This leaves the parameters to be estimated. This way the result (the original image RGB_r) is known, but the α, the original shadowless image ($RGB_{\hat{o}}$), and the overlay is not known (RGB_a). The new model looks like equation 13. There is now also an α for each pixel i. The overlay is actually a $RGBA$ image.

$$\begin{bmatrix} R_i \\ G_i \\ B_i \end{bmatrix}_r = \alpha_i \begin{bmatrix} R_i \\ G_i \\ B_i \end{bmatrix}_a + (1-\alpha_i) \begin{bmatrix} R_i \\ G_i \\ B_i \end{bmatrix}_{\hat{o}} \tag{13}$$

In an α-estimation one α and RGB_a is tested while $RGB_{\hat{o}}$ which should be the shadowless image of RGB_o is computed using equation 14.

$$\begin{bmatrix} R_i \\ G_i \\ B_i \end{bmatrix}_{\hat{o}} = \frac{1}{1-\alpha_i} \left(\begin{bmatrix} R_i \\ G_i \\ B_i \end{bmatrix}_r - \alpha_i \begin{bmatrix} R_i \\ G_i \\ B_i \end{bmatrix}_a \right) \tag{14}$$

The notation of the following investigation of optimal overlay color ($O = \{o_r, o_g, o_b\}$) will be simplified. The surface color (albedo) will be denoted S. The light from the sky will be E_{sky} and the light from the sun will be E_{sun}. We consider a sunlit pixel to be $S(E_{sky} + E_{sun})$ and an umbra pixel to be SE_{sky}. Furthermore, ambient occlusion and sunlight direction is assumed to be fixed.

$$\begin{aligned} SE_{sky} &= \alpha O + (1-\alpha)S(E_{sky} + E_{sun}) \\ \alpha O &= SE_{sky} - (1-\alpha)SE_{sky} - (1-\alpha)SE_{sun} \\ O &= \frac{(1-(1-\alpha))SE_{sky} - (1-\alpha)SE_{sun}}{\alpha} \\ O &= S(E_{sky} - \frac{(1-\alpha)}{\alpha}E_{sun}) \end{aligned} \tag{15}$$

We see that the optimal overlay color depends on the albedo, which is undesirable, as that means that the overlay color is not fixed. The albedo, the irradiance from the sun and the sky must be known in order to estimate α.

Consider when $\alpha = 1$ then the overlay color should be the actual umbra pixel. So if we choose the maximum α to be 1, then the color of the shadow free image is completely overwritten by the overlay, which means that the overlay should have the exact color of the surface in shadow. Hence, the maximum α in umbra should be as small is possible.

Consider $\alpha = 0.5$ then the overlay is $S(E_{sky} - E_{sun})$ which would probably be negative.

Consider $\lim \alpha \to 0^+$ then E_{sun} is weighted by $\lim_{\alpha \to 0^+}^{\frac{1-0^+}{0^+}} = \infty$. So if we choose the maximum α to be very small, the overlay will be large negative values. In order to avoid negative number, α should be as high as possible.

In order to minimize errors with a fixed overlay to be used for any surface albedo, the overlay could be calculated from $S = 50\%$.

Figure 3 shows how the model reacts to $\alpha = \{0, 0.1, ..., 1\}$ to two given surface colors and a given fixed overlay color. It shows that a fixed overlay is for from accurate. However, in the small *alpha* range ($\alpha < 0.5$), the direction is approximately linear.

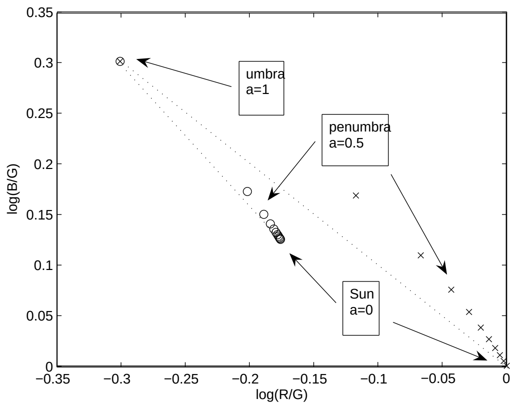


Fig. 3. The effect of different alphas on the log chromaticity plot on two surfaces with different albedos. The model does not approximate Finlaysons's model of shadows as an illumination direction as a straight line.

2.1.2 The multiplicity model

This model takes advantage of the relation described in equation 11. It is adapted to control the degree of shadow with α in equation 16 (for each pixel i).

$$\begin{bmatrix} R_i \\ G_i \\ B_i \end{bmatrix}_r = \begin{bmatrix} (1-\alpha_i O_r)R_i \\ (1-\alpha_i O_g)G_i \\ (1-\alpha_i O_b)B_i \end{bmatrix}_{\hat{o}} \tag{16}$$

The shadowless image can be calculated from the original image and an estimated α.

$$\begin{bmatrix} R_i \\ G_i \\ B_i \end{bmatrix}_{\hat{o}} = \begin{bmatrix} (1-\alpha_i O_r)^{-1}R_i \\ (1-\alpha_i O_g)^{-1}G_i \\ (1-\alpha_i O_b)^{-1}B_i \end{bmatrix}_r \tag{17}$$

The optimal overlay color is easy (using the short notation like before):

$$\begin{aligned} SE_{sky} &= (1-\alpha O)S(E_{sky}+E_{sun}) \qquad (18)\\ O &= \frac{(1-\frac{SE_{sky}}{S(E_{sky}+E_{sun})})}{\alpha_{max}} = \frac{(1-\frac{E_{sky}}{E_{sky}+E_{sun}})}{\alpha_{max}} \end{aligned}$$

α_{max} can be selected arbitrarily. The simplest would be to use $\alpha_{max} = 1$.

Note that the evidence needed to compute O is a sunlit pixel divided by its corresponding umbra pixel ($\frac{SE_{sky}}{S(E_{sky}+E_{sun})}$).

Consider $\alpha = 0$ then the sunlit pixel is weighted by 1, i.e. no change.

Consider $\alpha = 0.5 * \alpha_{max}$ then the color moves gradually toward umbra.

Consider $\alpha = \alpha_{max}$ then the sunlit pixel is weighted by the intended ratio between sunlit pixels and umbra pixels.

Figure 4 shows how the model reacts to $\alpha = \{0, 0.1, ..., 1\}$ to two given surface colors and a given overlay color. The model moves the 2-d log chromaticity plot along an approximate straight line. The points are not evenly distributed, so $\alpha = 0.5$ is not halfway toward umbra.

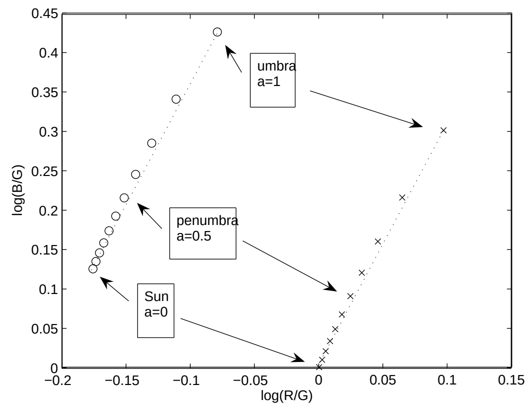


Fig. 4. The effect of different alphas on the log chromaticity plot on two albedos. The model approximates Finlaysons's model of shadows as a straight line.

3 Model Discussion

The additive model is difficult to use in practice because there are too many unknowns. Assuming a fixed overlay color the normal overlay model will always offset the chromaticity toward a fixed color (figure 3). Theoretically if this color is "infinitely" far away from the sunlit pixel color then it approximates an offset in a fixed direction. The model relates more intuitively to the physical model which is additive.

On the other hand the multiplicity model will offset it in a fixed direction. Furthermore, the overlay color in the multiplicity model is easy to compute from a known match of shadow pixels and sun pixels. The multiplicity model is also less complex.

Figure 5 shows two manually augmented examples using the additive model and the multiplicity model. The image is taken outdoor on a table. Notice that the red object reflects red light on the white object and while the sun comes from the left, there is a window to the right that reflects sun light. The overlay color for the multiplicity is chosen by measuring ratio between the red object in shadow and in sun. The artificial shadows are brushed in the alpha layer in Adobe Photoshop, so that the edges are soft.

It was harder to find a good overlay color for the additive model. It had to be as dark as possible but it could not be negative. Consequently, α was rather high (about 0.67). It was found be comparing some augmented shadow pixels to real shadow pixels on the same surface.

Table 1 shows that the chosen overlay for the multiplicity was the best augmentation.

4 Conclusion

Our contribution is two novel models of shadows such that natural shadows can be segmented and augmented virtual objects can cast exact soft shadows. The new feature in these models are variable penumbra (half-shadows). These have been analyzed analytically in relation to color theory and tangibility. The additive model is hard to estimate, while the multiplicity model is consistent with known color theory and easy to use.

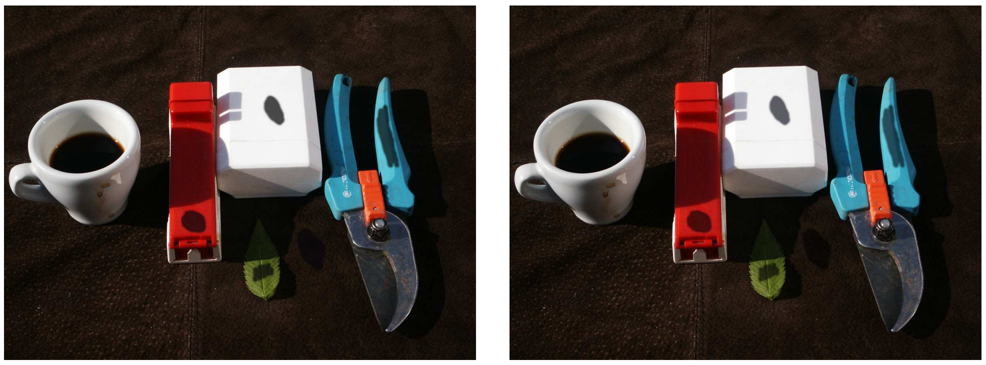


Fig. 5. Examples of fake shadows superimposed using the additive overlay (left) and the multiplicity overlay (right).

Table 1
Average RGB values in shadow regions. The multiplicity model maintains the actual chromaticities better. Sum of Squared difference errors of the log chromaticities for additive model is 1.1935 and for the multiplicity model it is 0.4212.

	Real	Additive	Multiplicity
Red R,G,B	74,9,10	60,12,12	74,11,10
Log Chr. r,b	2.1068 , 0.1054	1.6094 , 0	1.9062 ,-0.0953
Green R,G,B	20,26,10	30,39,16	33,37,14
Log Chr. r,b	-0.2624 , -0.9555	-0.2624 , -0.8910	-0.1144 , -0.9719
Blue R,G,B	20,67,95	32,62,70	36,77,92
Log Chr. r,b	-1.2090 , 0.3492	-0.6614 , 0.1214	-0.7603 , 0.1780
White R,G,B	103,92,106	77,77,78	100,100,108
Log Chr. r,b	0.1129 , 0.1417	0 , 0.0129	0 , 0.0770
Dark skin R,G,B	7,8,5	11,8,9	10,9,5
Log Chr. r,b	-0.1335 , -0.4700	0.3185 , 0.1178	0.1054 , -0.5878

Further work includes experiments with the color models in changing illumination. This will reveal if the effect of ambient occlusion and to some degree inter-reflections can be approximated by the models. A main element of the research is Graph cut (Kolmogorov and Zabih, 2002) segmentation of the shadows using the overlay models.

Acknowledgments

This research is funded by the CoSPE project (26-04-0171) under the Danish Research Agency. This support is gratefully acknowledged.

References

Barnard, K., Finlayson, G. D., 2000. Shadow identification using colour ratios. In: Color Imaging Conference. IS&T - The Society for Imaging Science and Technology, pp. 97–101.

Barnard, K., Funt, B., 1998. Experiments in sensor sharpening for color constancy. In: IS&T/SID Sixth Color Imaging Conference: Color Science, Systems and Applications Scottsdale, Arizona,November 1998. pp. 43–46.

Finlayson, G. D., Hordley, S. D., Drew, M. S., 2002a. Removing shadows from images. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (Eds.), ECCV (4). Vol. 2353 of Lecture Notes in Computer Science. Springer, pp. 823–836.

Finlayson, G. D., Hordley, S. D., Drew, M. S., 2002b. Removing shadows from images using retinex. In: Color Imaging Conference. IS&T - The Society for Imaging Science and Technology, pp. 73–79.

Kolmogorov, V., Zabih, R., 2002. What energy functions can be minimized via graph cuts? In: ECCV (3). pp. 65–81.

Lu, C., Drew, M. S., 2005. Shadow segmentation and shadow-free chromaticity via markov random fields. In: IS&T/SID 13th Color Imaging Conference.

Madsen, C. B., 2003. Using real shadows to create virtual ones. In: Bigün, J., Gustavsson, T. (Eds.), SCIA. Vol. 2749 of Lecture Notes in Computer Science. Springer, pp. 820–827.

Salvador, E., Cavallaro, A., Ebrahimi, T., 2001. Shadow identification and classification using invariant color models. In: Proc. of IEEE Signal Processing Society International Conference on Acoustics, Speech, and Signal Processing (ICASSP-2001) Salt Lake City (Utah, USA), 7-11 May. pp. 1545–1548.

Salvador, E., Cavallaro, A., Ebrahimi, T., 2004. Cast shadow segmentation using invariant color features. Comput. Vis. Image Underst. 95 (2), 238–259.

Wang, J., Cohen, M. F., 2005. An iterative optimization approach for unified image segmentation and matting. In: ICCV. IEEE Computer Society, pp. 936–943.

Action Recognition using Motion Primitives

T.B. Moeslund, P. Fihl, M.B. Holte, T.B. Moeslund

Laboratory of Computer Vision and Media Technology
Aalborg University, Denmark
Email: tbm@cvmt.dk

Abstract. The number of potential applications has made automatic recognition of human actions a very active research area. Different approaches have been followed based on trajectories through some state space. In this paper we also model an action as a trajectory through a state space, but we represent the actions as a sequence of temporal isolated instances, denoted primitives. These primitives are each defined by four features extracted from motion images. The primitives are recognized in each frame based on a trained classifier resulting in a sequence of primitives. From this sequence we recognize different temporal actions using a probabilistic Edit Distance method. The method is tested on different actions with and without noise and the results show recognizing rates of 88.7% and 85.5%, respectively.

1 Introduction

Automatic recognition of human actions is a very active research area due to its numerous applications. As opposed to earlier the current trend is not as much on first reconstructing the human and the pose of his/her limbs and *then* do the recognition on the joint angle data, but rather to do the recognition directly on the image data, e.g., silhouette data [18] [17] or temporal templates[4] [1].

Common for these approaches is that they represent an action by image data from all frames constituting the action, e.g., by a trajectory through some state-space or a spatio-temporal volume. This means that the methods in general require that the applied image information can be extracted reliably in every single frame. In some situations this will not be possible and therefore a different type of approach has been suggested. Here an action is divided into a number of smaller temporal sequences, for example movemes [6], atomic movements [7], states [5], dynamic instants [13], examplars [11], behaviour units [9], and key-frames [8]. The general idea is that approaches based on finding smaller units will be less sensitive compared to approaches based on an entire sequence of information.

For some approaches the union of the units represents the entire temporal sequence, whereas for other approaches the units represent only a subset of the original sequence. In Rao *et al.* [13] dynamic hand gestures are recognized by searching a trajectory in 3D space (x and y-position of the hand, and time) for certain dynamic instants. Gonzalez *et al.* [8] look for key-frames for recognizing actions, like walking and running. Approaches where the entire trajectory (one action) is represented by a number of subsequences, are Barbic *et al.* [2] for full body motion, where probabilistic PCA is used for finding transitions between different behaviors, and Bettinger *et al.* [3] where likelihoods are used to separate a trajectory into sub-trajectories. These sub-trajectories are modeled by Gaussian distributions each corresponding to a temporal primitive.

2 Paper Content and System Design

In this paper we address action recognition using temporal instances (denoted primitives) that only represent a subset of the original sequence. That is, our aim is to recognize an action by recognizing only a few primitives as opposed to recognition based on the entire sequence (possibly divided into sub-trajectories).

Our approach is based on the fact that an action will always be associated with a movement, which will manifest itself as temporal changes in the image. So by measuring the temporal changes in the image the action can be inferred. We define primitives as temporal instances with a significant change and an action is defined as a set of primitives. This approach allows for handling corrupted input sequences and as we shall see, does not require the lengths, the start point, nor the end point to be known, which is the case in many other systems.

Measuring the temporal changes can be done in a number of ways. We aim at primitives that are as independent on the environment as possible. Therefore, we do not rely on figure-ground segmentation using methods like background subtraction or personalized models etc. Instead we define our primitives based on image subtracting. Image subtraction has the benefit that it measures the change in the image over time and can handle very large changes in the environment.

Concretely we represent our primitives by four features extracted from a motion-image (found by image subtraction). In each frame the primitive, if any, that best explains the observed data is identified. This leads to a discrete recognition problem since a video sequence will be converted into a string containing a sequence of symbols, each representing a primitive. After pruning the string a probabilistic Edit Distance classifier is applied to identify which action best describes the pruned string. The system is illustrated in figure 1. The actions that we focus on in this work are five one-arm gestures,

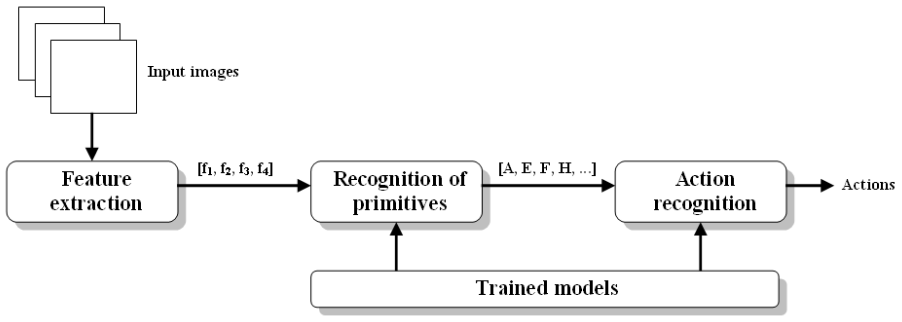


Fig. 1. System overview.

but the approach can with some modifications be generalized to body actions. The actions are inspired by [10] and can be seen in figure 2. The paper is structured as follows.

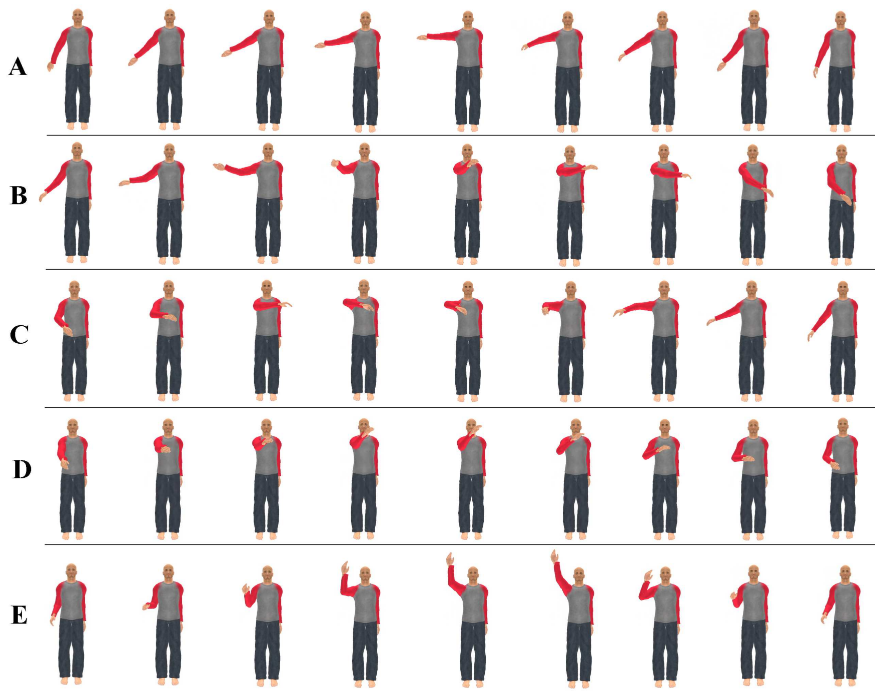


Fig. 2. Samples from the five actions. **A - Point right:** A stretched arm is raised to a horizontal position pointing right, and then lowered down. **B - Move left:** A stretched arm is raised to a horizontal position pointing right. The arm is then moved in front of the body ending at the right shoulder, and then lowered down. **C - Move right:** Right hand is moved up in front of the left shoulder. The arm is then stretched while moved all the way to the right, and then lowered down. **D - Move closer:** A stretched arm is raised to a horizontal position pointing forward while the palm is pointing upwards. The hand is then drawn to the chest, and lowered down. **E - Raise arm:** The arm is moved along the side of the person and stretched above the head, and then lowered again.

In section 3 we describe how our features are extracted. In section 4 we describe how we recognize the primitives, and in section 5 we describe how we recognize the actions. In section 6 the approach is evaluated on a number of actions and in section 7 the approach is discussed.

3 Feature Extraction

Even though image subtraction only provides crude information it has the benefit of being rather independent to illumination changes and clothing types and styles. Furthermore, no background model or person model is required. However, difference images suffer from ”shadow effects” and we therefore apply double difference images, which are known to be more robust [19]. The idea is to use three successive images in order to create two difference images. These are thresholded and ANDed together. This ensures that only pixels that have changed in both difference images are included in the final output. Multiple steps between the three successive images used to generate the double difference image have been tried out (frame 1-2-3, frame 1-3-5, and frame 1-4-7, etc.). The approach is rather invariant to this choice, i.e., invariant to the frame-rate and the execution speed of the actions. Frame 1-3-5 are used in this work.

When doing arm gestures the double difference image will roughly speaking contain a ”motion-cloud”. However, noise will also be present. Either from other movements, e.g., the clothes on the upper body when lifting the arm (false positives), or the motion-cloud will be split into a number of separate blobs, e.g., due to the shirt having a uniform color (false negatives). Since the two noise sources ”work against each other”, it is difficult to binarize the difference image. We therefore apply a hysteresis principle consisting of two thresholds T_1 and T_2 with $T_1 > T_2$. For all difference pixels above T_1 we initiate a region growing, which continues to grow until the pixel values falls below T_2, see figure 3. The resulting connected motion components are further sorted in respect to their size to obtain robustness towards noise. This hysteresis threshold helps to ensure that noisy motion-clouds are not broken up into multiple fragments and at the same time eliminates small noisy motion blobs. The result is one connected motion-cloud. We model the motion-cloud compactly by an ellipse. The length and orientation

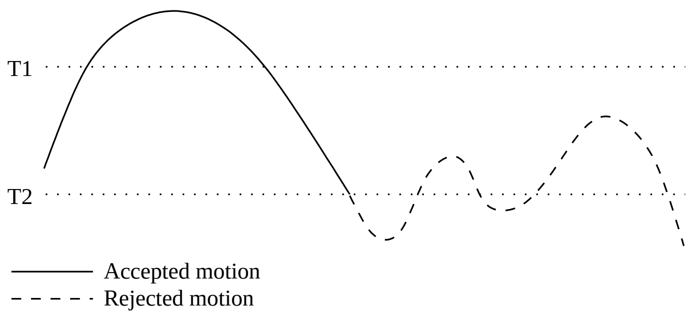


Fig. 3. An illustration of the hysteresis with an upper threshold T_1 and a lower threshold T_2. The figure illustrates the advantage of the hysteresis, where most of the ”motion-blob” of interest is accepted while the smaller ”noise-blobs” are rejected.

of the axes of the ellipse are calculated from the Eigen-vectors and Eigen-values of the covariance matrix defined by the motion pixels.

We use four features to represent this cloud. They are independent of image size and the person’s position in the image. Furthermore, two are defined with respect to a reference point currently defined manually as the center of gravity of the person. The features are: the eccentricity of the ellipse, the orientation of the ellipse, the size of the ellipse with respect to the distance from the reference point to the ellipse, and the angle between the reference point and the ellipse.

4 Recognition of Primitives

Each incoming frame is represented by the four extracted features described above. In this block the feature vector is classified as a particular primitive or noise. A Mahalanobis classifier is build by forming the covariance matrix for each primitive based on a set of representative training examples, see below. The four features are not equally important and therefore weighted in accordance with their importance. This yields the following classifier for recognizing a primitive at time, t:

$$\text{Primitive}(t) = \arg\min_i \left[(\boldsymbol{W} \cdot (\boldsymbol{f}_t - \boldsymbol{p}_i))^T \mathit{\Pi}_i^{-1} (\boldsymbol{W} \cdot (\boldsymbol{f}_t - \boldsymbol{p}_i)) \right] \quad (1)$$

where $\boldsymbol{f}_t$ is the feature vector estimated at time t, $\boldsymbol{p}_i$ is the mean vector of the ith primitive, $\mathit{\Pi}_i$ is the covariance matrix of the ith primitive, and $\boldsymbol{W}$ contains the weights and are included as an element-wise multiplication.

The classification of a sequence can be viewed as a trajectory through the 4D feature space where, at each time-step, the closest primitive (in terms of Mahalanobis distance) is found. To reduce noise in this process we introduce a minimum Mahalanobis distance in order for a primitive to be considered in the first place. Furthermore, to reduce the flickering observed when the trajectory passes through a border region between two primitives we introduce a hysteresis threshold. It favors the primitive recognized in the preceding frame over all other primitives by modifying the individual distances. The classifier hereby obtains a ”sticky” effect, which handles a large part of the flickering.

After processing a sequence the output will be a string with the same length as the sequence. An example is illustrated in equation 2. Each letter corresponds to a recognized primitive and Ø corresponds to time instances where no primitives are below the minimum required Mahalanobis distance. The string is pruned by first removing ’Ø’s, isolated instances, and then all repeated letters, see equation 3. A weight is generated to reflect the number of repeated letters (this is used below).

$$\text{String} = \{\emptyset, \emptyset, B, B, B, B, B, E, A, A, F, F, F, F, \emptyset, D, D, G, G, G, G, \emptyset\} \quad (2)$$

$$\text{String} = \{B, A, F, D, G\} \quad (3)$$

$$\text{Weights} = \{5, 2, 4, 2, 4\} \quad (4)$$

4.1 Learning Models for the Primitives

In order to recognize the primitives we need to have a prototypical representation of each primitive, i.e., a mean and covariance in the 4D feature space. As can be seen in figure 2 the actions are all fronto-parallel.

Ongoing work aims at generalizes this work by allow for multiple viewpoints. One problem with this is how to train the system - it will require a very large number of test sequences. Therefore we have captured all training data using a magnetic tracking system with four sensors. The sensor placements are: one at the wrist, one at the elbow, one at the shoulder, and one at the upper torso (for reference). The hardware used is the

Polhemus FastTrac [15] which gives a maximum sampling rate of 25Hz when using all four sensors. The data is converted into four Euler angles: three at the shoulder and one at the elbow in order to make the data invariant to body size. An action corresponds to a trajectory through a 4D space spanned by the Euler angles.

The data is input to a commercial computer graphics human model, Poser [16], which then animates all captured data. This allows us to generate training data for any view point and to generate additional training data by varying the Euler angles (based on the training data) and varying the clothing of the model. Figure 4 shows a person with magnetic trackers mounted on the arm, two different visualizations of the 3D tracker data from Poser, and an example of the test data. Based on this synthetic training data we build a classifier for each primitive.

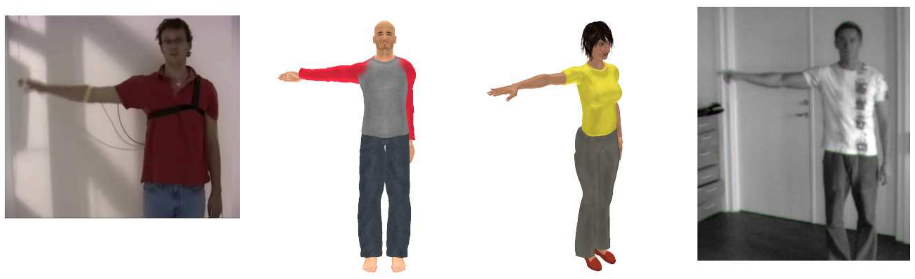


Fig. 4. An illustration of the different types of data used in the system. From left to right: 1) 3D tracker data is acquired from magnetic trackers mounted on persons who perform the five actions. 2) The tracker data is animated in Poser from a fronto-parallel view. 3) The tracker data can be animated from any view point with different clothings and models. 4) After training the primitives on semi-sythetic data we recognize actions in real video.

4.2 Defining the Primitives

Defining the number of primitives and their characteristics (”human movement”) is quite a significant optimization problem. We are aiming at automating this process [14], but in this work it was done manually.

The primitives are defined based on an evaluation of video sequences showing three different people performing the five actions. The criteria for defining the primitives are 1) that they represent characteristic and representative 3D configurations, 2) that their projected 2D configurations contain a certain amount of fronto-parallel motion, and 3) that the primitives are used in the description of as many actions as possible, i.e., fewer primitives are required. In this way we find 10 primitives that can represent the five actions. Each primitive is appearing in several actions resulting in five to eight primitives for each action.

To obtain the prototypical representation we randomly select 20 samples of each primitive from the training video sequences. The double difference images of these samples are calculated and the motion-clouds are each represented by the four features.

The 20 samples then yields a mean vector and a 4x4 covariance matrix for each primitive.

In figure 4 the 10 primitives and their representations are visualized together with the letter denoting the primitive.

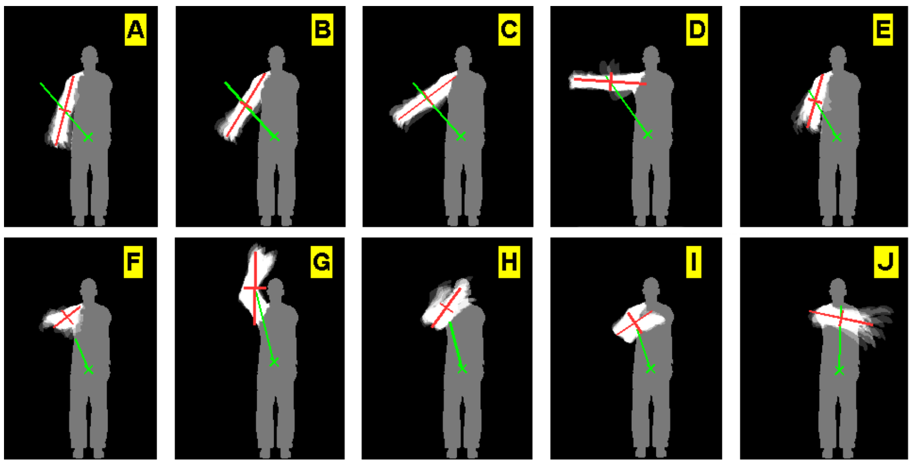


Fig. 5. The figure of each primitive contains the silhouettes of the 20 samples added together which gives the gray silhouette. The 20 motion clouds from the double difference images of the samples are added on top of the silhouette as the white cloud. The figures furthermore illustrates the mean of the four features for each primitive by depicting the axes of the fitted ellipse and the distance and direction from the reference point to the motion cloud.

5 Recognition of Actions

The result of recognizing the primitives is a string of letters referring to the known primitives. During a training phase a string representation of each action to be recognized is learned. The task is now to compare each of the learned actions (strings) with the detected string. Since the learned strings and the detected strings (possibly including errors!) will in general not have the same length, the standard pattern recognition methods will not suffice. We therefore apply the Edit Distance method [12], which can handle matching of strings of different lengths.

The edit distance is a well known method for comparing words or text strings, e.g., for spell-checking and plagiarism detection. It operates by measuring the distance between two strings in terms of the number of operations needed in order to transform one to the other. There are three possible operations: *insert* a letter from the other string, *delete* a letter, and *exchange* a letter by one from the other string. Whenever one of these operations is required in order to make the strings more similar, the score or distance is increased by one.

When the strings representing the actions are of different lengths, the method tends to favor the shorter strings. Say we have detected the string $\{B, C, D\}$ and want to classify it as being one of the two actions: $\#1 = \{J, C, G\}$ and $\#2 = \{A, B, C, D, H\}$. The edit distance from the detected string to the action-strings will be two in both cases. However, it seems more likely that the correct interpretation is that the detected string comes from action #2 in a situation where the start and end has been corrupted by noise. In fact, 2 out of 3 of the primitives have to be changed for action #1 whereas only 2 out of 5 have to be changed for action #2. We therefore normalize the edit distance by dividing the output by the length of the action-string, yielding 0.67 for action #1 and 0.2 for action #2, i.e., action #2 is recognized.

The edit distance is a deterministic method but by changing the cost of each of the three operations with respect to likelihoods it becomes a probabilistic method[1]. Concretely we apply the weights described above, see equation 4. These to some extent represent the likelihood of a certain primitive being correct. The higher the weight the more likely a primitive will be. We incorporate the weights into the edit distance method by increasing the score by the weight multiplied by β (a scaling factor) whenever a primitive is *deleted* or *exchanged*. The cost of *inserting* remains 1.

The above principle works for situations where the input sequence only contains one action (possibly corrupted by noise). In a real scenario, however, we will have sequences which are potentially much longer than an action and which might include more actions after each other. The action recognition problem is therefore formulated as for each action to find the substring in the detected string, which has the minimum edit distance. The recognized action will then be the one of the substrings with the minimum distance. Denoting the start point and length of the substring, s and l, respectively, we recognize the action present in the detected string as:

$$\text{Action} = \arg\min_{k,s,l} PED(\Lambda, k, s, l) \tag{5}$$

where k index the different actions, Λ is the detected string, and $PED(\cdot)$ is the probabilistic edit distance.

6 Results

6.1 Test Setup

Two kind of tests are conducted: one with known start and stop time of action execution, and another with ”noise” added in the beginning and end of the sequences (unknown start time). By adding noise to the sequence we introduce the realistic problem of having no clear idea about when an action commence and terminates which would be the case in a real situation. To achieve a test scenario that resembles this situation we split the five actions into halves and add one of these half actions randomly to the beginning and one to the end of each action to be processed by the system. In this way we get an unknown start and end point of the real action.

We use eleven test subjects, whom each performs each gesture 10 times. This leads to 550 sequences. The weighting of the features $\boldsymbol{W}$ are set to $\{1, 4, 2, 4\}$, and $\beta = 1/8$. A string representation of each action is found and since the shortest string contains five primitives and the longest eight primitives, we only perform the probabilistic edit distance calculation for substrings having the lengths $\in [3, 15]$.

[1] This is related to the Weighted Edit Distance method, which however has fixed weights.

6.2 Tests

The overall recognition rate for the test with known start time is 88.7%. In figure 6(a) the confusion matrix for the results is shown. As can be seen in the figure, most of the errors occur by miss-classification between the two actions: *move closer* and *raise arm*. The main reasons for this confusion are different performances by the test subjects (some do not raise their arm very much when preforming the *raise arm* action), the similarity of the actions, and the similarity of the primitives in these actions. As can be seen in figure 2 both actions are performed along the side of the person when seen from the fronto-parallel view and differs mainly in how high the arm is raised. From figure 5 it can be seen that primitives ’F’, ’G’, ’H’, and ’I’ have similar angles between the reference point and the motion cloud and ’F’, ’H’ and ’I’ also have similar orientation of the ellipse. These two features, which are the ones with highest weights, make these four primitives harder to distinguish.

Figure 6(b) shows the confusion matrix for the test results with noise. The overall recognition rate for this test is 85.5%, which is 3.2% lower than without noise. The errors are the same as before but with some few additional errors caused by the unknown start and end time of the actions.

	1.	2.	3.	4.	5.
1. Point right	**110**				
2. Move left	**7**	**100**		**3**	
3. Move right	**6**		**102**	**1**	**1**
4. Move closer		**3**	**2**	**78**	**26**
5. Raise arm				**12**	**98**

(a) Known start and stop time.

	1.	2.	3.	4.	5.
1. Point right	**109**		**1**		
2. Move left	**10**	**99**		**1**	
3. Move right	**8**		**99**	**3**	
4. Move closer	**1**	**5**	**2**	**69**	**33**
5. Raise arm	**2**	**2**		**12**	**94**

(b) Unknown start and stop time.

Fig. 6. The confusion matrix for the recognition of the different actions with and without noise.

7 Conclusion

In this paper we have presented an action recognition approach based on motion primitives as opposed to trajectories. Furthermore, we extract features from temporally local motion as opposed to background subtraction or another segmentation method relying on learned models and a relatively controlled environment. We hope this makes our approach less sensitive, but have still to prove so in a more comprehensive test.

The results are promising due to two facts. First, the models are generated from synthetic data (generated based on test subjects) while the test data are real data. In fact, the test data and training data are recorded several months apart, hence this is a real test of the generalization capabilities of the action recognition process. This means that we can expect to use the same scheme when learning models for the next incarnation of the system, which is aimed at view-invariant action recognition. Secondly, the system does not break down when exposed to realistic noise. This suggests that the approach taking has potential to be expanded into a real system setup, as opposed to a lab setup which is virtually always used when testing action recognition systems.

The primitives used in this work are found manually. This turned out to be quite an effort due to the massive amount of data and possibilities. Currently we are therefore working to automate this process [14]. Another ongoing activity is to avoid manually defining the reference point, see section 3, by using the face (found by an Adaboost trained face detector) as a reference for the features.

References

1. R.V. Babu and K.R. Ramakrishnan. Compressed domain human motion recognition using motion history information. In *Proc. Int. Conf. on Acoustics, Speech and Signal Processing*, Hong Kong, April 6-10, 2003.
2. J. Barbic, N.S. Pollard, J.K. Hodgins, C. Faloutsos, J-Y. Pan, and A. Safonova. Segmenting Motion Capture Data into Distinct Behaviors. In *Graphics Interface*, London, Ontario, Canada, May 17-19 2004.
3. F. Bettinger and T.F. Cootes. A Model of Facial Behaviour. In *IEEE International Conference on Automatic Face and Gesture Recognition*, Seoul, Korea, May 17 - 19 2004.
4. A. Bobick and J. Davis. The Recognition of Human Movement Using Temporal Templates. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 23(3), 2001.
5. A.F. Bobick and J. Davis. A Statebased Approach to the Representation and Recognition of Gestures. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(12), 1997.
6. C. Bregler. Learning and Recognizing Human Dynamics in Video Sequences. In *Conference on Computer Vision and Pattern Recognition*, pages 568 – 574, San Juan, Puerto Rico, 1997.
7. L. Campbell and A. Bobick. Recognition of Human Body Motion Using Phase Space Constraints. In *International Conference on Computer Vision*, Cambridge, Massachusetts, 1995.
8. J. Gonzalez, J. Varona, F.X. Roca, and J.J. Villanueva. *aSpaces*: Action spaces for recognition and synthesis of human actions. In *AMDO*, pages 189–200, AMDO02, 2002.
9. O.C. Jenkins and M.J. Mataric. Deriving Action and Behavior Primitives from Human Motion Data. In *Proc. IEEE Int. Conf. on Intelligent Robots and Systems*, pages 2551–2556, Lausanne, Switzerland, Sept.30 – Oct.4, 2002.
10. A. Just and S. Marcel. HMM and IOHMM for the Recognition of Mono- and Bi-Manual 3D Hand Gestures. In *ICPR workshop on Visual Observation of Deictic Gestures (POINTING04)*, Cambridge, UK, August 2004.
11. A. Kale, N. Cuntoor, and R. Chellappa. A Framework for Activity-Specific Human Recognition. In *International Conference on Acoustics, Speech and Signal Processing*, Orlando, Florida, May 2002.
12. V.I. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Doklady Akademii Nauk SSSR*, 163(4):845–848, 1965.
13. C. Rao, A. Yilmaz, and M. Shah. View-Invariant Representation and Recognition of Actions. *Journal of Computer Vision*, 50(2):55 – 63, 2002.
14. L. Reng, T.B. Moeslund, and E. Granum. Finding Motion Primitives in Human Body Gestures. In S. Gibet, N. Courty, and J.-F. Kamps, editors, *GW 2005*, number 3881 in LNAI, pages 133–144. Springer Berlin Heidelberg, 2006.
15. *http://polhemus.com/*, January 2006.
16. *http://www.poserworld.com/*, January 2006.
17. D. Weinberg, R. Ronfard, and E. Boyer. Motion History Volumes for Free Viewpoint Action Recognition. In *IEEE Int. Workshop on Modeling People and Human Interaction*, 2005.
18. A. Yilmaz and M. Shah. Actions Sketch: A Novel Action Representation. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, San Diego, CA, June 20-25, 2005.
19. K. Yoshinari and M. Michihito. A Human Motion Estimation Method using 3-Successive Video Frames. In *Int. Conf. on Virtual Systems and Multimedia*, Gifu, Japan, 1996.

Sideways Stepping

Bo Markussen[*], Jon Sporring[‡], and Kenny Erleben[‡]

[*] bomar@kvl.dk

The Royal Veterinary
and Acricultural University
Bülowsvej 17
DK-1870 Frederiksberg C
Denmark

[‡] {sporring,kenny}@diku.dk

Department of Computer Science
University of Copenhagen
Universitetsparken 1
DK-2100 Copenhagen
Denmark

July, 2006

Abstract

We analyse vector fields, where only the normal component is known. Examples of such fields are: optical flow fields and warps of signed distance maps. We propose to model the tangential component by minimizing an general energy functional of the total field, and we present a novel iterative solution based on Euler-Lagrange equations. Possible applications are estimating physical flow in image sequences, estimating human growth processes, and co-warping textures in animation sequences.

Scan Conversion of Signed Distance Fields

Kenny Erleben and Henrik Dohlmann*

Department of Computer Science, University of Copenhagen, Denmark

Abstract

Fast and robust signed distance field computation is often either a performance bottleneck due to high resolution fields or nearly impossible due to degeneracies in input meshes. Thus, it can be tedious and very time consuming to obtain a signed distance field to be used for collision detection in for instance a physical based animation, motion planing, or geometry processing.

Sign leaking problems in scan conversion methods may result in erroneous signed distance fields for even perfect two-manifold meshes. We present solutions for the sign leaking problems. The major contribution is the robust handling of errors caused by overlapping bounded volumes of neighboring features.

Our method is simple to implement, and has a tradeoff between performance and quality. Further the method is robust in the sense that it handles the sign problems of previous work.

The novelty lies in representing the narrow-band as a decomposition of tetrahedra, a shell mesh. We provide numerous examples and comparisons on different methods for generating the narrow-band shell.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation;

Keywords: Tetrahedra Mesh, GPGPU, Scan Conversion, Distance Field

1 Introduction

Signed distance fields are very attractive in computer graphics and related fields. Often they are used for collision detection in cloth animation [Bridson et al. 2003], multibody dynamics [Guendelman et al. 2003], deformable objects [Fisher and Lin 2001], also mesh generation [Molino et al. 2003; Persson and Strang 2004], motion planning [Hoff, III et al. 1999], and sculpting [Bærentzen 2001].

In all of these applications a signed distance field is represented as a regular sampling of the closest distance to the surface of an object. Usually the convention of using negative values inside the object and positive values outside the object is applied. There does exist adaptive distance fields [Frisken et al. 2000], but we will not consider these in this paper.

The first problem of using distance fields is to actually compute them. There are several issues involved here: The object surface is in most cases modeled as a polygonal model by an animator or obtained by some means of scanning or segmentation. In all cases one often has to deal with a polygonal model having holes, flipped surfaces, overlapping faces, and much worse. This is often termed inconsistent meshes [Bischoff et al. 2005]. Inconsistent meshes are unpleasant, mostly because it is not always meaningful, what is inside or outside.

The computational complexity is also often problematic. A naïve implementation on CPU can take hours, even days, to complete for high-resolution grids (256^3 resolution or greater).

Thus the practicalities in obtaining a signed distance field is often overwhelming, and it is these problems that is the focus of this paper.

The brute force approach to computing distance fields can be described as: For each grid node compute the closest distance to the faces in a polygonal model. Acceleration techniques do exist, such as only querying grid nodes against a bounding volume hierarchy or reversing the iteration to iterate over bounding volumes around faces. These previous methods used a two-pass strategy to resolve the sign issue. In [Aanæs and Bærentzen 2003] angle weighted pseudo-normals was used to determine the correct sign, thus allowing for a single pass only.

A straightforward parallelization of the naïve approach is possible by reversing the order of iteration, that is for each face compute the distance to all grid nodes. This was done in [Hoff, III et al. 1999]. Here the authors mesh the distance function of a vertex, edge, or face, and render it directly to the depth buffer. For volumes this is done in a slice by slice manner, and the distance field is read back from the depth buffer. Any distance metric can be used, but signs are not handled. The simplicity of the method is attractive, although it requires tessellation of elliptical cones and hyperboloid sheets in 3D. Obviously the tessellation causes discretization errors in the distance computation, but the errors can be controlled. This approach is henceforth termed distance meshing.

Scan conversion algorithms using the GPU have become quite popular. Here various external regions is scan converted which bounds the space of points lying closer to a geometric feature, than any other geometric feature. These methods require the construction of bounded volumes that is scan-converted in a slice by slice manner. For each grid node being rendered (voxel), a distance value is being computed. In [Mauch 2003] the characteristic scan conversion (CSC) algorithm was presented. Here three different kinds of Characteristic Polyhedra is used: A prism (for faces), a cone (for vertices), and a wedge (for edges). Conceptually easy to understand it is not very clear how the curved surfaces of the cones and wedges should be tessellated. To avoid aliasing, the polyhedra was enlarged, however the author did not describe the possible errors in the computations, caused by grid nodes getting caught on the wrong side of the surface. This artifact is described in detail in Section 2.

In [Sigg et al. 2003] an optimized GPU version of CSC is presented, together with a more aggressive scan-conversion method, named Prism Scan. Here prisms are constructed for faces only, thus reducing the number of bounded volumes that need to be scan converted. Also a novel fragment program is presented for computing the signed distances of the rasterized grid nodes.

Prism Scan suffers from the same sign problems as CSC, since only the face plane are used to determine the sign, explained in de-

*{kenny,henrikd}@diku.dk

tail in Section 2. These sign errors may seem very innocent since they only occur rarely for small narrow-bands and smooth curved objects. However if narrow-band size is increased and objects with sharp ridges and valleys are scan converted, the sign errors immediately blows up as huge areas of discontinuities where the wrong side of the surface is leaked into the other side.

Both Prism Scan and CSC are limited by a user specified narrow-band size, unlike the distance meshing approach which is capable of computing a full grid.

Both these methods relies on the input surface mesh to be a perfect two-manifold. Working with real-world models this is often not the case and one must often resort to some kind of mesh reconstruction [Nooruddin and Turk 2003]. In this paper we will take the stand point that meshes might be ugly and may cause errors in the signed distance field.

In [Sud et al. 2004] several performance improvements for computing distance fields on graphics hardware are presented. The main two contributions is a culling method based on occlusion queries and a conservative clamping computation based on the spatial coherency of the distance field. Although distance meshing was used in this paper, the method generalizes to scan-conversion algorithms as well, here the conservative clamping can be used to control the size of the narrow-band parameter.

To summarize, methods for computing distance fields on graphics hardware falls into two different approaches: distance meshing or scan conversion of bounded volumes. In [Hsieh and Tai 2005] a hybrid of these two approaches is presented for the 2D case.

Other approaches involve solving the Eikonal equation using for instance a two stage fast marching method [Sethian 1999b]. First one marches from the surface out, then from the surface in. In order to be efficient these methods rely on a good (fast) heap implementation. Besides, one have to seed the fast marching method by computing the distance values on nodes lying just next to the surface. Whereas the scan conversion methods described above compute exact signed distance fields, fast marching methods have a discretization error of order $O(1)$. Although [Sethian 1999a] presents a higher order accurate version.

There are other ways for dealing with the computation of distance fields, such as Danielsson's distance field algorithm [Danielsson 1980]. Here a four pass scan method is used to propagate distance information on a regular 2D grid. The method can be extended to 3D, and has some resemblance with the fast marching method.

We will not discuss CPU based algorithms any further since it is our goal to exploit graphics hardware to obtain a sufficient performance.

In this paper we will present a novel scan conversion approach. Our approach combines the novel fragment program from Prism Scan with the pseudo-normal method. Hereby, we avoid any sign errors of the previous scan conversion algorithms.

Our approach uses a tetrahedral shell [Erleben and Dohlmann 2004; Erleben et al. 2005], meaning only tetrahedra volumes are considered. This allows the usage of a fast tetrahedron slicer to compute cross sections. Slicing tetrahedra are well known from volume visualization and are extremely efficient. Thus it is cheaper to compute cross-sections from scratch, and render these, than doing a 3D scan conversion of more complex prisms, cones, or wedges.

We will present and discuss several methods for computing tetrahedra shells.

- First we will discuss a shell generation method based on simple extrusion along vertex angle weighted pseudo normals, which will be combined with a convex hull computation [Barber et al. 1996], and an enlargement to handle aliasing. The convex hull computation will ensure consistency and removal of degeneracies.

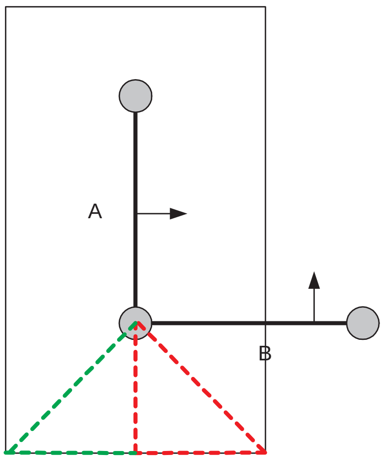


Figure 1: Plane test sign error. The figure shows a cross section of a polygonal model focused on two faces, A and B. Shown together with the bounded region around A, in which the signed distance is calculated. In the dashed red area, the points are closest to A, and the sign will therefore become positive. This is clearly wrong. The points in the dashed red region is located inside the object, so the sign should have been negative. The sign will be correct in the green dashed region. If the planes alone are used to determine the distance, then the distance will be wrong in both the red and green dashed areas.

- Then we will extend the simple method with the extrusion algorithm from the thin tetrahedral shell mesh [Erleben and Dohlmann 2004] combined with a simple face normal extrusion technique to avoid leaking. The benefit over the more simple approach is a more tight fitting shell with less overlapping tetrahedra, thus implying fewer rasterized grid nodes.

- Finally an oriented bounding box (OBB) fitting method is presented, which is simple and easy to implement. In comparison with the other methods, it may have large overlapping regions, even regions expanding far beyond the wanted narrow-band.

The first two shell creation methods, we present, rely on the ability to compute the angle weighted vertex pseudo normals, the last shell creation method makes no assumption on the mesh whatsoever and can be used for unstructured meshes with all kinds of degeneracies.

Note any kind of tetrahedral shell generation method could be used in our scan conversion method, e.g. the adaptive thin tetrahedral shell mesh [Erleben et al. 2005]. This allow for a tradeoff between simplicity of creation and efficiency of scan-conversion.

The correct sign computation in the fragment program relies on the angle weighted pseudo normals of both vertices and edges. If these cannot be computed correctly, then there is no guarantee that the method will compute the proper sign of the distance field.

We have organized our paper as follows: In Section 2 we describe the leaking problems and their sources. Hereafter we present our method in Section 3 and our results in Section 4. Finally we conclude in Section 5.

2 Leaking

In characteristic scan conversion (CSC) and Prism Scan a plane test is used to determine the sign of the distance function, as illustrated in Figure 1. As seen in the figure, this may lead to incorrect computation of the sign. In the case of CSC this becomes even worse, because the characteristic polyhedra are enlarged to avoid aliasing.

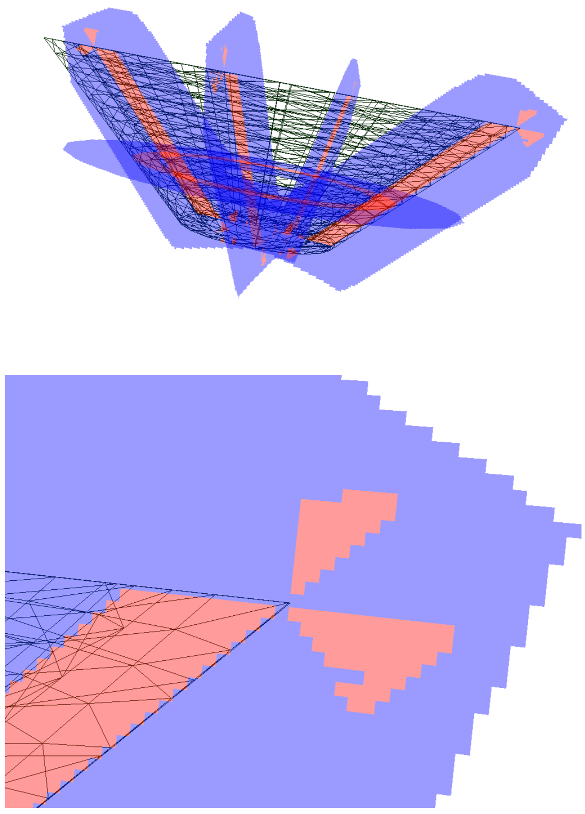


Figure 2: Real life example of leaking due to the plane test problem. Note that the red color have leaked into the blue color. Red is negative and blue is positive. Figure 7 shows the result using our method.

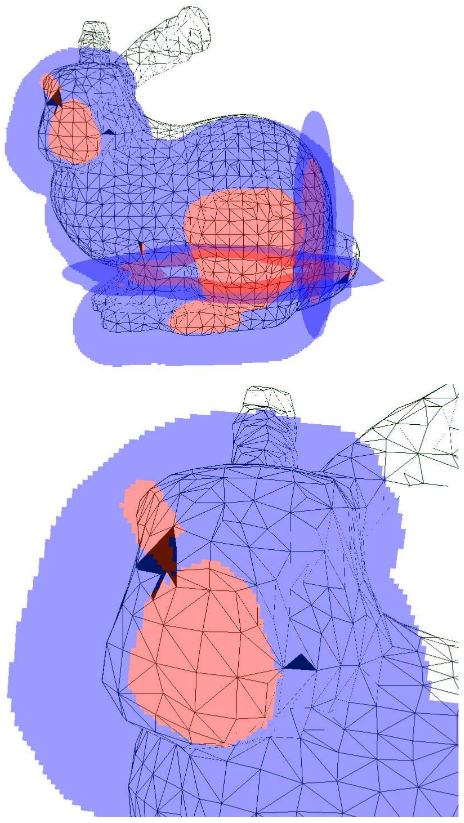


Figure 3: Real life example of leaking by the folding problem. Note that red color is leaking into the blue color. Red is negative and blue is positive. The triangles shaded in black are folded.

Thus for the face case, the distance of grid nodes outside the face-Voronoi region are also computed wrt. the face plane. This means that the dashed lines will produce grid nodes with distances close to zero inside Voronoi regions of neighboring faces. Prism Scan performs a case analysis of grid nodes in enlarged regions and will only suffer from a wrong sign computation. Figure 2 show real life examples of these problems.

Requiring a mesh to be a two-manifold is not a sufficient condition to avoid sign problems. If a surface mesh contains folds, then the orientation of a triangle face can be flipped. Thus, if scan-conversion algorithms are used, then the results depend on the scan order. If the flipped face is scan converted first, then it will result in wrong sign computations. This creates a strange leaking effect. Figure 4 illustrates the mesh-topology of a fold, and Figure 3 shows a real life example.

The final source for leaking problems is due to construction of bounding volumes representing the narrow-band. This is illustrated in Figure 5. Here, the narrow-band shells have different widths on opposite sides of a thin region. This causes the inside region of one side to extend beyond the outside region on the opposite side. In Figure 6 a real life example is shown.

Our method the Tetrahedra (T4) GPU scan method is capable of handling the leaking by plane test and construction problems as shown in Figure 7.

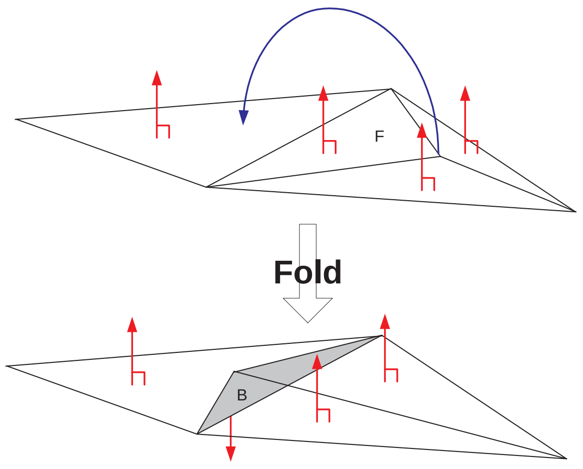


Figure 4: A planar mesh is folded, such that the back side B of a triangle is turning outside instead of the front side F.

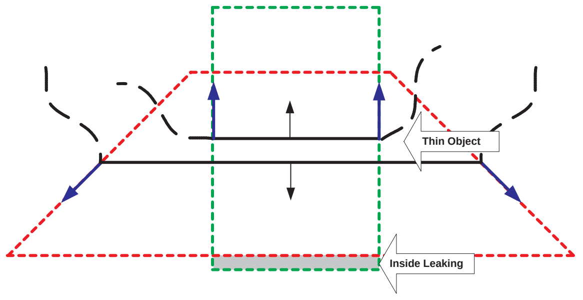


Figure 5: Cross section of a mesh with thin structure, shown together with their bounding regions. The top face has a bounding region shown in green, and the bottom face has a bounding region shown in red. For this configuration, the bounding region of the top face extrudes below the bounding region of the bottom face. This results in the small grey area, wherein the distance becomes negative.

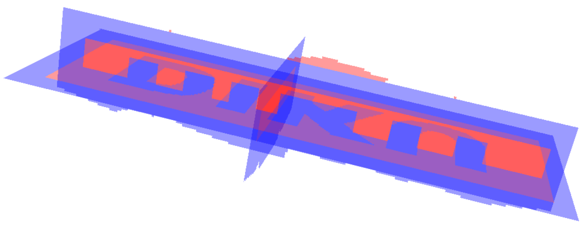


Figure 6: Real life example of leaking by construction problem. Observe the red area on the wrong side of the blue area. Figure 7 shows the result using our method.

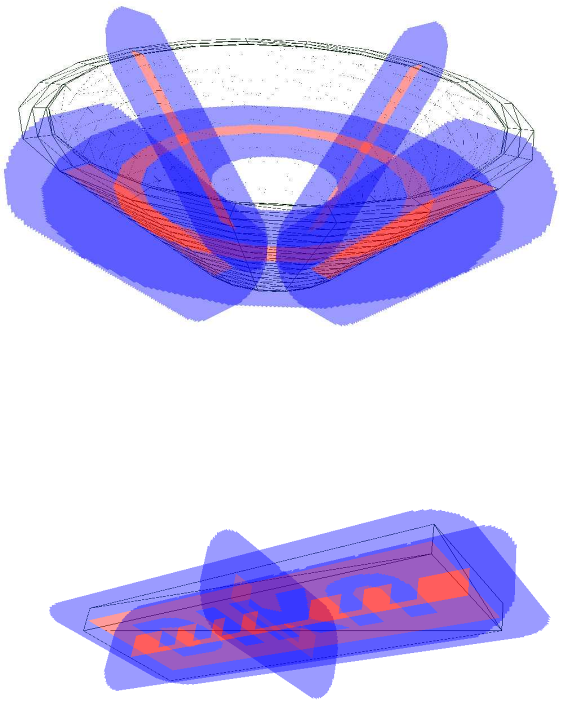


Figure 7: Example showing how our T4 GPU scan method handles the real-life examples from Figure 2 and 6. Notice that no leaking is present.

3 The T4 GPU Scan Method

We call our method the tetrahedron (T4) GPU scan method. In the following we will give an overview of our method.

Given a surface of an object as a collection of triangles, we compute the signed distance within a user specified narrow-band. First we generate tetrahedra in such a way that a single tetrahedron is related to a single triangle face. Note that several tetrahedra can be related to the same triangle face. Tetrahedra are generated while iterating over the triangular faces. Figure 8 illustrates the tetrahedra shell creation in pseudo code. A tetrahedron bounds a region of space containing a subset of grid nodes. For each of these grid nodes, the closest distance to the related triangle face is computed, and the sign is determined using pseudo-normals.

In order to determine the grid nodes lying inside a tetrahedron we move a z-plane in the direction of the positive z-axis. At each z-slice of the regular grid, we halt the z-plane and find the cross sections between tetrahedra and the z-plane. We have adopted a simple sweep-line [de Berg et al. 1997] algorithm to quickly find all tetrahedra that intersects the z-plane. As an alternative one could use the occlusion query method from [Sud et al. 2004].

Having found the cross-sections we render these and use a GPU fragment program to compute the signed distances. Before moving on to the next z-slice of the regular grid, we read back the computed distance values from the frame-buffer, and store it in an internal data structure.

Figure 9 shows the overall steps of the T4 GPU scan method. Note that the shell creation could be done during the scan conversion, which will minimize storage usage. However, in our implementation we have chosen to keep the shell creation as a separate stage for better modularity of the implementation.

Our shell creation methods presented in Section 3.3, 3.4, and 3.5 have linear time complexity, $O(n)$ in the number of triangle faces n, because while iterating once over the triangle faces a fixed number of tetrahedra is generated for each triangle face. The initialization of the sweep-line ie. the z-sorting of the tetrahedra have $O(nlgn)$ time complexity, although the actual scan-conversion can be expected to have linear complexity in the number of generated tetrahedra.

In the following subsections we will describe the details of the individual steps. In Section 3.1 we describe an efficient method for computing the cross-section of a tetrahedron and a z-plane. In Section 3.2 we describe how to compute the signed distance values in a fragment program. In section 3.3 we describe a simple approach to shell creation, which is extended in Section 3.4. Finally in Section 3.5 another shell creation approach is described.

3.1 Computing Cross Section of a Tetrahedron

We use tetrahedra as the underlying primitive that bounds the mesh. To scan convert the distance field, we therefore need an efficient

```
algorithm create-shell()
  L = empty list
  for each face F
    generate tetrahedra of F
    add tetrahedra to L
  next F
  sort L in increasing min. z-values
end algorithm
```

Figure 8: Pseudo code for tetrahedra shell generation.

```
algorithm T4-GPU-scan()
  for z = min z plane to max z plane
    set S = {t in L, and intersects z}
    for tetrahedra t in S
      find cross-section with z
      render cross section
    next t
    read back distance values
  next z
end algorithm
```

Figure 9: Pseudo code for our tetrahedra (T4) GPU scan conversion method.

way to slice a tetrahedron with a plane. This method is inspired by [Bærentzen 2005].

To calculate a cross section of a tetrahedron, the four points of the tetrahedron is sorted by increasing z-value. This allows for a very simple algorithm to find the number of intersections and to create polygons to be processed by the fragment program.

Consider Figure 10. If the z-plane under consideration is below the lowest point in the tetrahedron, then there will be no intersections. Similar, if the z-plane is above the highest point in the tetrahedron, then there will be no intersections.

There are only three topologically distinct ways a z-plane can actually slice the tetrahedron:

A: The z-plane lies below p_1. In this case the plane cuts the lines p_0p_3, p_0p_1, and p_0p_2.

B: The z-plane lies between p_1 and p_2. In this case the plane cuts the lines p_0p_3, p_1p_3, p_1p_2, and p_0p_2.

C: The z-plane lies above p_2. In this case the plane cuts the lines p_0p_3, p_1p_3, and p_2p_3.

In case B, the polygon will always be convex. This can be seen by drawing all the possible configurations of a tetrahedron and consider the order, in which the plane cuts the four lines.

Cases, where the tetrahedron is only sliced in one point or along a line, has no area and should not be considered. The above algorithm ensures this never happens.

The polygons might be either clockwise or counter-clockwise, so a post-process might be necessary to ensure a proper orientation. However, the T4 GPU Scan method does not need this property.

3.2 Computing the Sign using Angle Weighted Pseudo Normals

A novel fragment program was introduced in [Sigg et al. 2003], which calculated the distance to a triangle. Here we give a description of the case analysis used to determine the distance, together

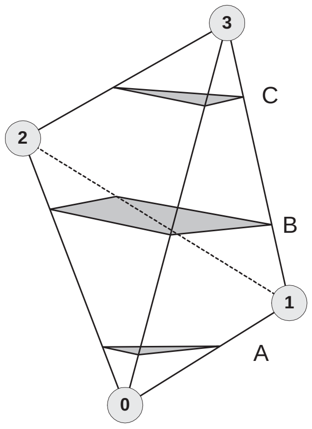


Figure 10: The possible topological different slicings of a tetrahedron.

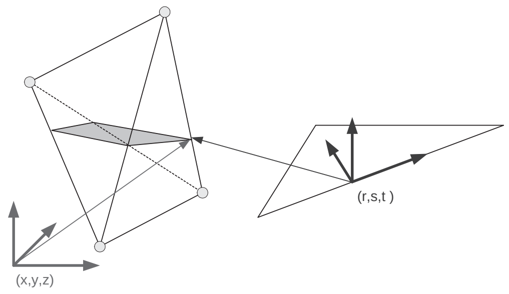


Figure 11: Local triangle frame for a triangle in the mesh and a related cross section.

with our extension that calculates the correct sign using the angle weighted pseudo normals.

Each triangle on the mesh is encased in a bounded volume. Each bounding volume consists of tetrahedra. The triangle is used to create a local triangle frame consisting of vectors $\vec{r}$, $\vec{s}$ and $\vec{t}$, as shown in Figure 11. The coordinates of the slice of the tetrahedron is converted to the local triangle frame and send to the GPU as texture coordinates.

The triangle on the mesh is analyzed to produce three lengths: the height called h, the length from the origin of the triangle frame to vertex $\vec{v}_1$ called a, and the length from the origin to vertex $\vec{v}_0$ called b. See Figure 12. Further, the six angle weighted pseudo normals, $\vec{n}_{v_0}$, $\vec{n}_{v_1}$, and $\vec{n}_{v_2}$ for the vertices, and $\vec{n}_{e_0}$, $\vec{n}_{e_1}$, and $\vec{n}_{e_2}$ for the edges, are calculated and transformed to the local triangle frame using a rotation matrix constructed from unit column vectors, as shown in (1).

$$\vec{n}' = \begin{bmatrix} \frac{\vec{a}}{||\vec{a}||} & \frac{\vec{h}}{||\vec{h}||} & \frac{\vec{n}}{||\vec{n}||} \end{bmatrix}^T \vec{n}, \tag{1}$$

where $\vec{n}'$ is the transformed normal of $\vec{n}$. These pseudo normals and the three lengths are sent to the GPU as texture coordinates.

On the GPU, the first thing that happens is a reduction of the problem to the half-plane, where $r \geq 0$. That is, if the r-coordinate is negative, we flip the data such that $r = -r$, $a = b$, $\vec{n}_{v_1} = \vec{n}_{v_0}$, and $\vec{n}_{e_1} = \vec{n}_{e_2}$. This reduces the further analysis considerably.

Next, the r'-, and s'-coordinates is constructed from the r- and s-coordinates, and a case analysis is performed according to regions shown in Figure 13. From the case analysis, the distance to the closest feature can be computed, and the corresponding pseudo-normal can be determined. The sign of point $\vec{p}$ can be computed using the pseudo normal of the closest feature, $\vec{n}(\vec{c})$, and some point, $\vec{c}$, on

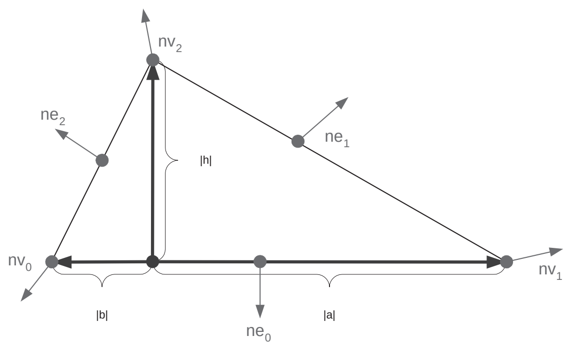


Figure 12: Local triangle with lengths a, b and h, and pseudo normals.

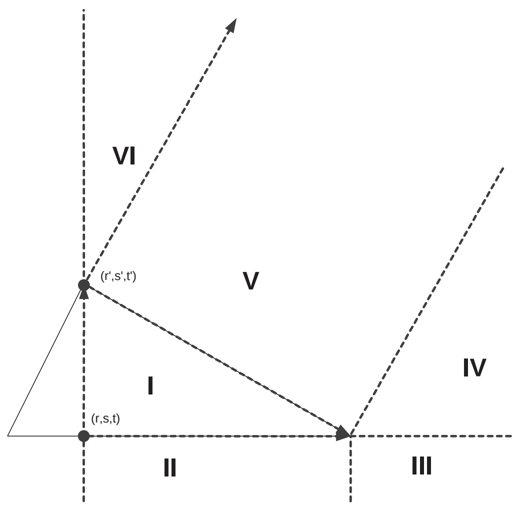


Figure 13: Regions used in the case analysis for the triangle.

the closest feature, as

$$d = \vec{n}(\vec{c}) \cdot (\vec{p} - \vec{c}), \tag{2}$$

as described in [Bærentzen and Aanæs 2005].

3.3 Angle Weighted Vertex Pseudo Normal Shell

For each triangular face we extrude the user-specified narrow-band distance, ε, outward and inward along vertex normals, which generates 6 points. Then we compute the convex hull to get a convex mesh completely covering and enclosing the triangular face, as shown in Figure 14. Hereafter we use the center of the convex mesh as apex for each generated tetrahedron and the triangular faces of the convex hull as bases of the generated tetrahedra. If non-triangular faces are found, then we use a simple ear-clipping algorithm [O'Rourke 1998] to tessellate these into triangular faces.

Note that this will in general not generate a connected tetrahedra mesh. In some cases tetrahedra generated from one face do not connect nicely with tetrahedra generated from neighboring faces.

This is either because, the quadrilateral faces of the generated prism in Figure 14 are not necessarily planar, or because the extrusion may cross over and create a swallow tail as shown in Figure 15.

In both cases using the convex hull of the extruded vertices ensures a conservative coverage; it also guarantees that no gaps will occur between the bounded region of the convex hull and the

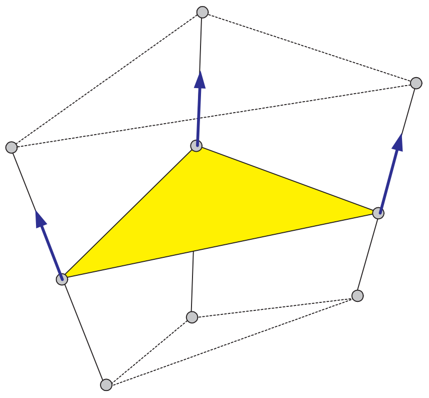


Figure 14: Convex hull of pseudo normal extruded vertices.

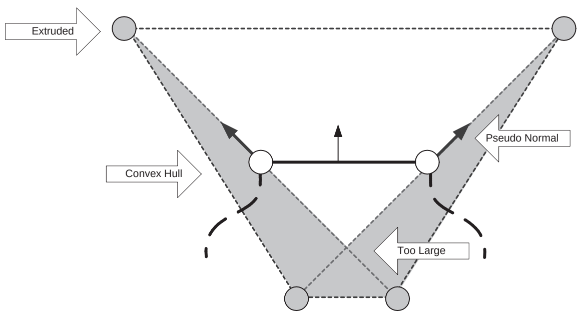


Figure 15: Swallow-tail extrusion making shell region of a single face too large. The too large region is illustrated in grey.

bounded regions of hulls from neighboring faces. The drawback is that grid nodes belonging to overlapping regions will be scan-converted more than once, causing a slight performance degradation.

Aliasing artifacts from rasterization of the sliced cross-sections may cause empty voxels inside the narrow-band. Working with floating point arithmetics may lead to numerical imprecision and truncation errors. Thus, even if the tetrahedra generated from two neighboring faces are perfectly meeting along the shared edge of the faces, then truncation and imprecision can lead to small voids. Furthermore, obscure faces could result in oblong tetrahedra, which would generate slivers when rendered. In conclusion, empty regions are unavoidable unless we do something extra.

The method of choice in the past have been to enlarge the polyhedra being scan-converted. However past methods did not recognize the leaking problems caused by the enlargement. Besides, enlargement have an inherent scale dependency of mesh size versus grid spacing, thus leading to an element of parameter tuning. Conservative rasterization [Hasselgren et al. 2005] ensures no aliasing effects and have no element of parameter tuning. The only drawback is a computational penalty, due to extra geometry processing in the vertex program pipeline. Another problem with the pseudo normal based shell creation method is that it may result in a leaking problem, due to the way the shell mesh is constructed.

3.4 Thin Tetrahedral Shell

The shell creation method in the previous section suffers from the swallow tail problem. This causes the shell around a single face too become too large, thus causing large overlapping regions with neighboring faces. This is illustrated in Figure 15. To remedy

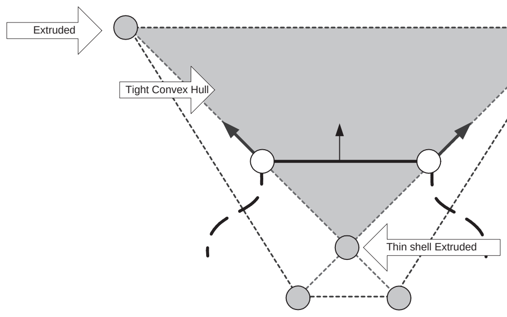


Figure 16: Using thin-shell extrusion lengths result in a more tight fitting convex hull, illustrated by the grey area.

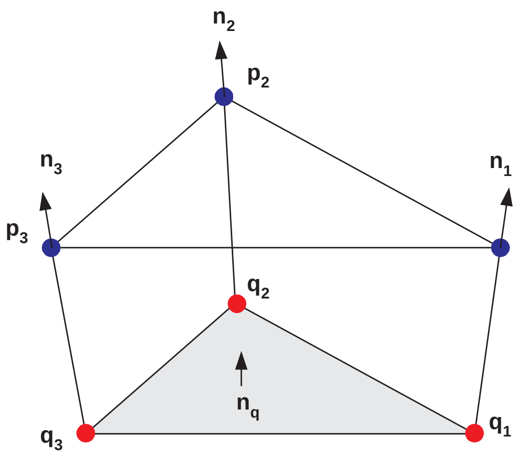


Figure 17: The six corner points defining a prism, and vectors yielding extrusion directions.

this problem we could try to keep the shell region of a single face as tight as possible, e.g. by using the thin-shell extrusion length method in [Erleben and Dohlmann 2004], the details of which will be described later.

Using the thin-shell extrusion length method we compute three inward extruded vertices and three outward extruded vertices, for a total of 6 vertex positions which is passed along to the convex hull algorithm before doing the tetrahedra tessellation. Again convex hull and apex construction method can be applied as in the pseudo normal shell method. Note that if the extrusion length is limited, then the coordinates of some of the 6 extruded vertices will be the same. Figure 16 illustrates the thin-shell idea.

Given a triangle consisting of three vertices $\vec{p}_1$, $\vec{p}_2$, and $\vec{p}_3$, with corresponding three unit direction vectors (we use the angle weighted normals) $\vec{n}_1$, $\vec{n}_2$, and $\vec{n}_3$, indicating the extrusion line direction, Then the inward extruded prism is defined by the six corner points $(\vec{p}_1, \vec{p}_2, \vec{p}_3)$ and $(\vec{q}_1, \vec{q}_2, \vec{q}_3)$ where:

$$\vec{q}_1(\varepsilon) = \vec{p}_1 - \vec{n}_1\varepsilon \tag{3}$$
$$\vec{q}_2(\varepsilon) = \vec{p}_2 - \vec{n}_2\varepsilon \tag{4}$$
$$\vec{q}_3(\varepsilon) = \vec{p}_3 - \vec{n}_3\varepsilon. \tag{5}$$

The extrusion length is given by $\varepsilon > 0$. Notation is illustrated in Figure 17. Similarly, the corner points of the outward extrusion can be found by flipping the extrusion direction vectors.

By requiring ε to be strictly positive, all generated prisms will have non-zero volume. We therefore seek a robust way to determine an upper bound on ε, such that the prism will be valid.

The direction of the normal of the extruded face, $\vec{n}_q$, can be found from $\vec{q}_1$, $\vec{q}_2$, and $\vec{q}_3$, using the cross-product:

$$\vec{n}_q(\varepsilon) = (\vec{q}_2(\varepsilon) - \vec{q}_1(\varepsilon)) \times (\vec{q}_3(\varepsilon) - \vec{q}_1(\varepsilon)). \tag{6}$$

This is a second order polynomial in ε,

$$\vec{n}_q(\varepsilon) = \vec{a}\varepsilon^2 + \vec{b}\varepsilon + \vec{c}, \tag{7}$$

where

$$\vec{a} = (\vec{n}_1 - \vec{n}_2) \times (\vec{n}_1 - \vec{n}_3) \tag{8}$$
$$\vec{b} = (\vec{p}_2 - \vec{p}_1) \times (\vec{n}_1 - \vec{n}_3) + (\vec{n}_1 - \vec{n}_2) \times (\vec{p}_3 - \vec{p}_1) \tag{9}$$
$$\vec{c} = (\vec{p}_2 - \vec{p}_1) \times (\vec{p}_3 - \vec{p}_1). \tag{10}$$

Observe that $\vec{c} \neq \vec{0}$, since its magnitude is equal to twice the area of the triangle being extruded.

To ensure we avoid a swallow-tail, the dot product of the direction of the normal of the extruded face, $\vec{n}_q$, with the vectors, $\vec{n}_1$, $\vec{n}_2$, and $\vec{n}_3$, must always be positive. That is $\vec{n}_1 \cdot \vec{n}_q(\varepsilon) > 0$, $\vec{n}_2 \cdot \vec{n}_q(\varepsilon) > 0$, and $\vec{n}_3 \cdot \vec{n}_q(\varepsilon) > 0$. This yields the following system of constraints,

$$\begin{bmatrix} \vec{n}_1 \cdot \vec{a} & \vec{n}_1 \cdot \vec{b} & \vec{n}_1 \cdot \vec{c} \\ \vec{n}_2 \cdot \vec{a} & \vec{n}_2 \cdot \vec{b} & \vec{n}_2 \cdot \vec{c} \\ \vec{n}_3 \cdot \vec{a} & \vec{n}_3 \cdot \vec{b} & \vec{n}_3 \cdot \vec{c} \end{bmatrix} \begin{bmatrix} \varepsilon^2 \\ \varepsilon \\ 1 \end{bmatrix} > 0. \tag{11}$$

We solve for the smallest positive ε fulfilling the system of constraints. That is, each row represents the coefficient of a second order polynomial in ε, thus for each row we find the two roots of the corresponding polynomial. The three rows yields a total of 6 roots. If no positive root exist, then $\varepsilon = \infty$, otherwise ε is set equal to the smallest positive root.

In fact, the tree dot-product constraints ensure that no neighboring prism will intersect each other, nor will the prism turn its inside out (ie. flipping the extruded face opposite the original face).

The thin-shell extrusion length creation method can lead to a leaking artifact, when creating shells for thin objects. This is illustrated in Figure 5. Computing extrusion lengths along vertex normals only guarantee that we reach the outer boundary of the narrow-band along the vertex normals. Everywhere else the computed narrow-band will have less extent than along the vertex normals.

To minimize the chance of leaking due to differences in pseudo normal angles and trying to make the narrow-band evenly thick, we could extend the current shell creation method with more extruded vertices. We have chosen to make an outward and inward extrusion of the face vertices along the face normal, thus passing a total of 12 extruded points to the convex hull algorithm. We term this heuristic "face-offsetting".

The idea of face-offsetting is illustrated in Figure 18. Face-offsetting do result in lesser tight shell region around the face. Thus increasing overlap with shell region of neighboring faces. This causes more grid nodes to be scan converted.

3.5 OBB Shell

Using the longest edge, $\vec{e}$, and the orthogonal height vector, $\vec{h}$, a tight fitting rectangle can be placed in the face plane of the triangle. Hereafter the rectangle is enlarged by the user-specified narrow-band size, ε. Finally the four vertices of the rectangle are extruded ε-distance outward and inward along the face normal, $\vec{n}$, in order to produce an enclosing OBB around the triangle face. Figure 19 illustrates the steps involved. The OBB can be directly decomposed into 5 tetrahedra. This is very simple to implement and nearly impossible to get wrong. It does ensure a complete coverage of the narrow-band, although large parts may stick outside or overlap. Thus simplicity comes at a performance degradation.

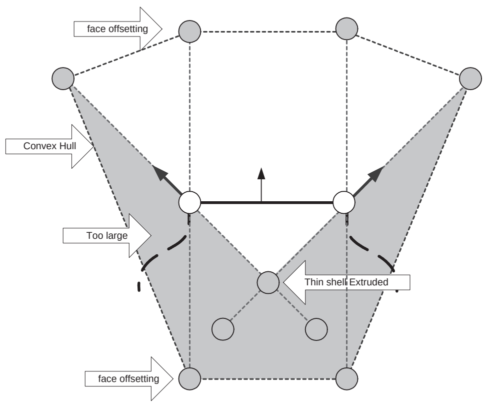


Figure 18: Using face-offsetting to minimize chance of leaking and creating a more evenly thick narrow-band. Grey area show the redundant region.

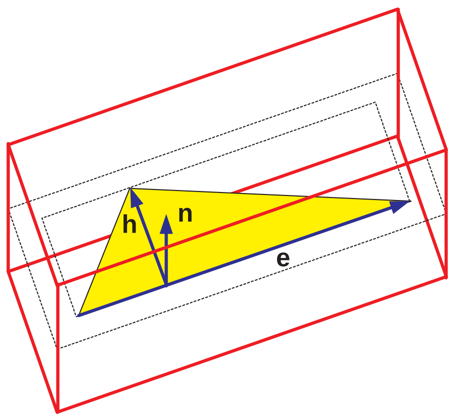


Figure 19: Fitting an OBB around a triangle face.

4 Results

All measurements were performed on a 2.4 GHz P4, with 4GB RAM, running Gentoo Linux. The graphics card installed is a Geforce 6800GT with 256MB RAM. We used a narrow-band size corresponding to 10% of the maximum mesh extend.

We have plotted performance measurements in Figures 20, 21, 22, 23, 24, 25, and 26. As expected all figures shows linear complexity that scales with mesh sizes.

Figures 22, 23, 24, and 25 show the CPU time overhead for the 4 different configurations. A clear bottleneck in our implementation is the lookup operations in our tetrahedra mesh. Next most expensive operation is surface mesh lookup of vertex coordinates and normals.

Figure 26 shows the GPU time overhead. It shows that the fragment program is computationally most expensive and out-weights the frame-buffer read back for large mesh sizes.

In Figure 27 we have shown a few of our signed distance field results using OBB shell creation method. Left column shows the sign computation. Middle column shown the signed distance field. Right column has the mesh super-imposed. Note that no leaking is present, and that the signed distance field appears smooth everywhere.

Figure 28 shows the different narrow-bands obtained using the different shell creation methods. Here it is clearly seen that pseudo-normal extrusion, thin-shell extrusion, and face-offsetting creates a

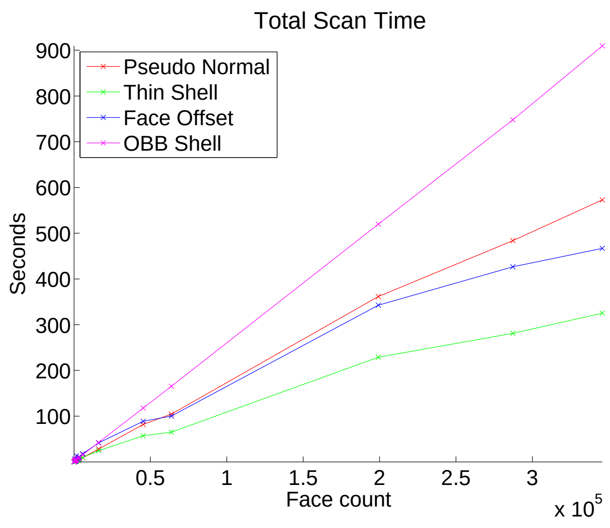


Figure 20: Time used totally.

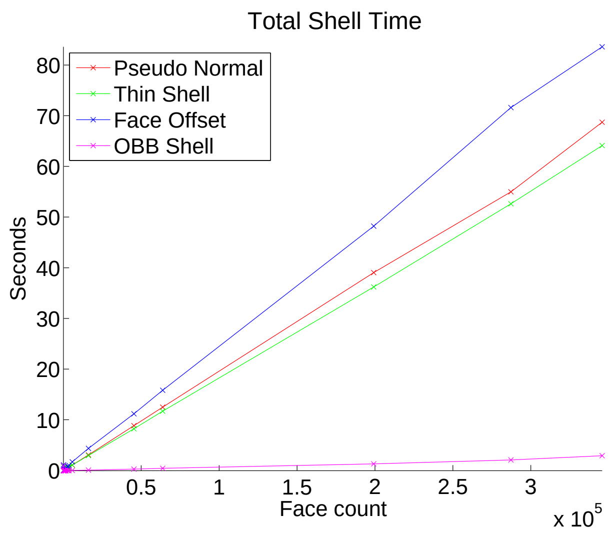


Figure 21: Time used to create the shell.

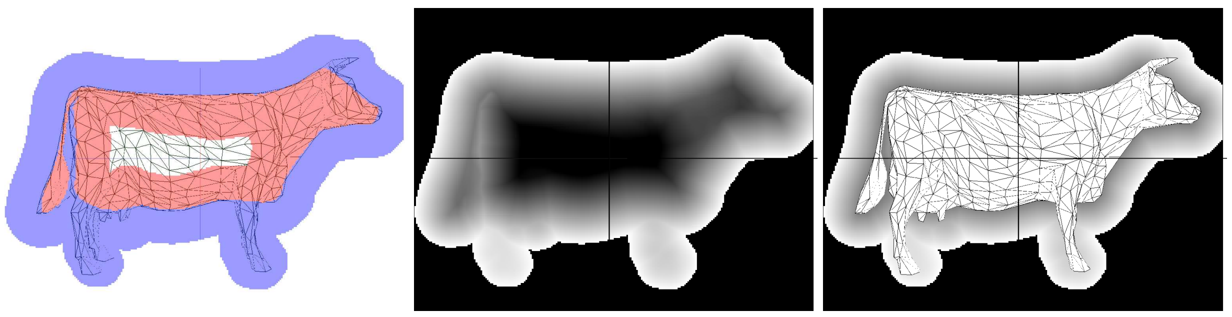
somewhat jagged narrow-band. The OBB creation method clearly yields the best quality, however it is also the one with worst scan conversion performance as seen in Figure 20. This is due to the too large OBBs extending far beyond the narrow-band size and having large overlaps.

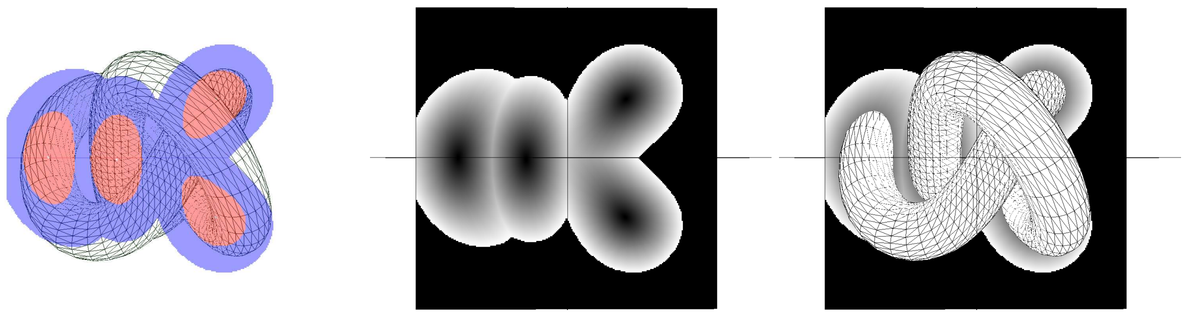
5 Conclusion

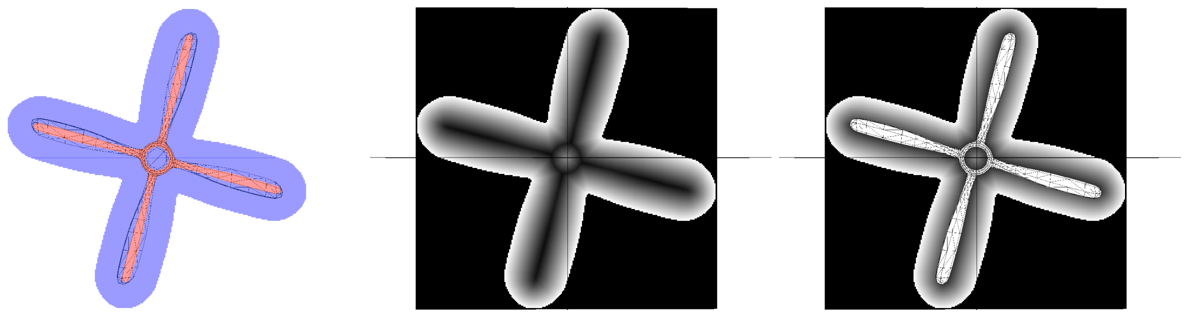
We have presented an approach for scan conversion of signed distance fields.

- It is based on a single type of simple geometry: a tetrahedron.
- It uses pseudo normals to handle correct sign computations.

We have presented several shell generation methods and discussed drawbacks and benefits. They are all simple to understand easy to implement. All put together our work yields a robust, simple, and efficient system for computing signed distance fields.


(a) Cow


(b) Knot


(c) Propeller

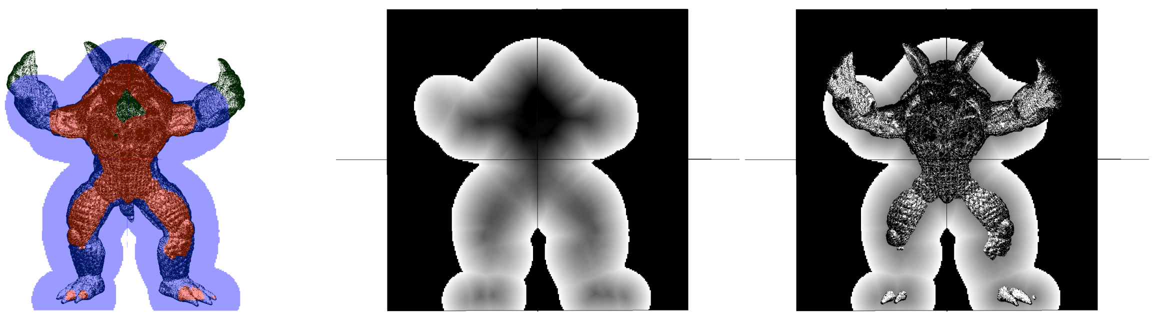
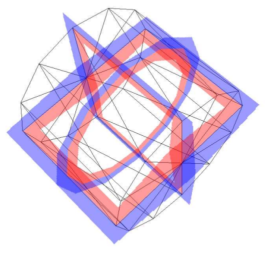
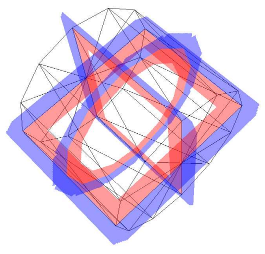

(d) armadillo

Figure 27: Sign verification and Signed Distance Field Results. Red is negative and blue is positive.


(a) Pseudo normal


(b) Thin shell with face offset

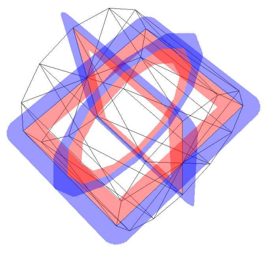

(c) OBB shell

Figure 28: Differences in shell creation method illustrated using a cylinder. Red is negative and blue is positive.

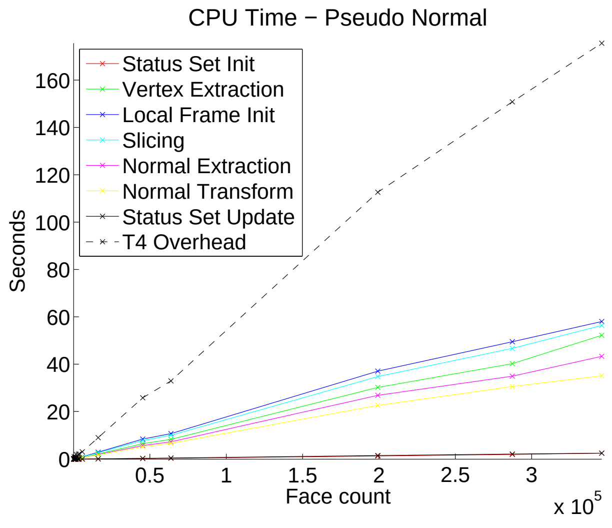


Figure 22: Time used to process the geometry on the CPU for pseudo normal extrusion and anti-aliasing.

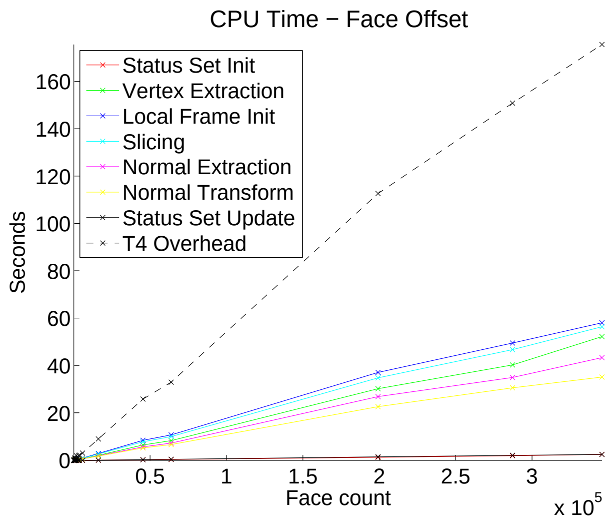


Figure 24: Time used to process the geometry on the CPU for the addition of face offsetting.

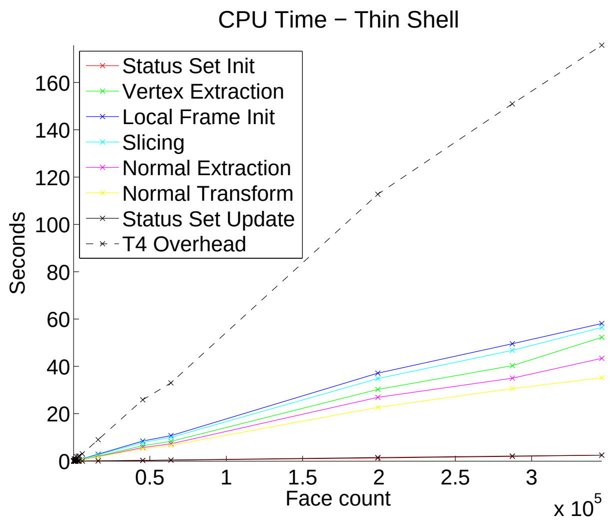


Figure 23: Time used to process the geometry on the CPU for thin shell extrusion length.

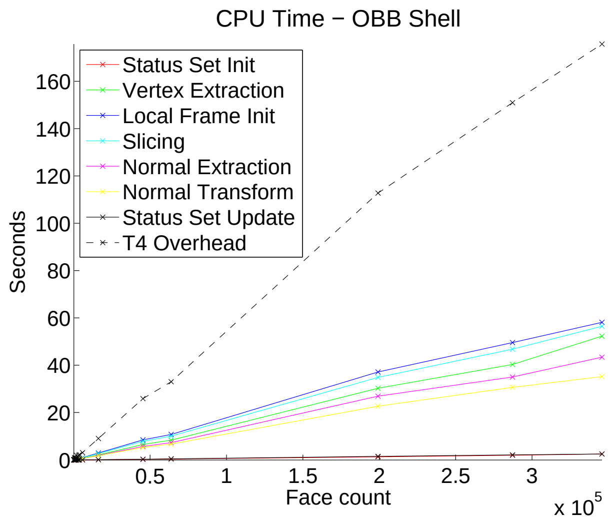


Figure 25: Time used to process the geometry on the CPU for the OBB shell creation method.

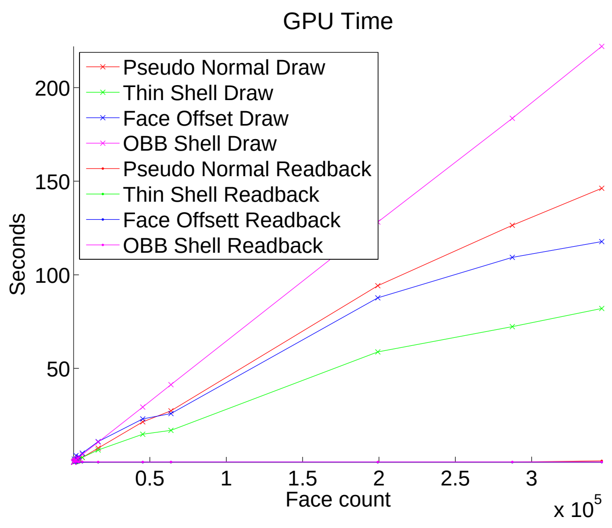


Figure 26: Time used to process the geometry on the GPU.

There is still room for improvements in this work. Faster methods of generating tight fitting tetrahedral shell meshes could boost the performance.

Although we have presented a shell generation method not relying on pseudo normals, the fragment program does need the pseudo normals. This may be an disadvantage for several degenerate meshes, that have redundant vertices creating open boundaries, which meet, but are not topologically connected. Other than that, the method is capable of handling open boundaries, even overlapping faces. Future work could focus on the dependence on pseudo normals, for instance by an algorithm capable of computing meaningful pseudo normals for degenerate meshes.

Besides, we have shown that folding cannot be handled with pseudo-normals. We speculate that a solution to folding problems requires a two-pass method. This is left as future work.

References

AANÆS, H., AND BÆRENTZEN, J. A. 2003. Pseudo–normals for signed distance computation. In *Proceedings of VISION, MODELING, AND VISUALIZATION*.

BÆRENTZEN, J. A., AND AANÆS, H. 2005. Signed distance computation using the angle weighted pseudo-normal. *Transactions on Visualization and Computer Graphics 11*, 3 (June), 243–253.

BÆRENTZEN, J. A. 2001. *Manipulation of Volumetric Solids, with application to sculpting*. PhD thesis, IMM, Technical University of Denmark. BMP 08-0011-311.

BÆRENTZEN, J. A., 2005. Personal communication.

BARBER, C. B., DOBKIN, D. P., AND HUHDANPAA, H. 1996. The quickhull algorithm for convex hulls. *ACM Trans. Math. Softw. 22*, 4, 469–483.

BISCHOFF, S., PAVIC, D., AND KOBBELT, L. 2005. Automatic restoration of polygon models. *ACM Trans. Graph. 24*, 4, 1332–1352.

BRIDSON, R., MARINO, S., AND FEDKIW, R. 2003. Simulation of clothing with folds and wrinkles. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Eurographics Association, 28–36.

DANIELSSON, P. E. 1980. Euclidean distance mapping. *Computer Graphics and Image Processing 14*, 227–248.

DE BERG, M., VAN KREVELD, M., OVERMARS, M., AND SCHWARZKOPF, O. 1997. *Computational Geometry, Algorithms and Applications*. Springer-Verlag.

ERLEBEN, K., AND DOHLMANN, H. 2004. The thin shell tetrahedral mesh. In *Proceedings of DSAGM*, S. I. Olsen, Ed., 94–102.

ERLEBEN, K., DOHLMANN, H., AND SPORRING, J. 2005. The adaptive thin shell tetrahedral mesh. *Journal of WSCG*, 17–24.

FISHER, S., AND LIN, M. C. 2001. Deformed distance fields for simulation of non-penetrating flexible bodies. In *Proceedings of the Eurographic workshop on Computer animation and simulation*, Springer-Verlag New York, Inc., 99–111.

FRISKEN, S. F., PERRY, R. N., ROCKWOOD, A. P., AND JONES, T. R. 2000. Adaptively sampled distance fields: a general representation of shape for computer graphics. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., 249–254.

GUENDELMAN, E., BRIDSON, R., AND FEDKIW, R. 2003. Nonconvex rigid bodies with stacking. *ACM Transaction on Graphics, Proceedings of ACM SIGGRAPH*.

HASSELGREN, J., AKENINE-MÖLLER, T., AND OHLSSON, L. 2005. *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation*. NVIDIA Corporation, ch. Conservative Rasterization on the GPU.

HOFF, III, K. E., KEYSER, J., LIN, M., MANOCHA, D., AND CULVER, T. 1999. Fast computation of generalized voronoi diagrams using graphics hardware. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., 277–286.

HSIEH, H.-H., AND TAI, W.-K. 2005. A simple gpu-based approach for 3d voronoi diagram construction and visualization. *Simulation Modelling Practice and Theory 13*, 8, 681–692.

MAUCH, S. 2003. *Efficient Algorithms for Solving Static Hamilton-JAcobi Equations*. PhD thesis, California Institue of Technology.

MOLINO, N., BRIDSON, R., TERAN, J., AND FEDKIW, R. 2003. A crystalline, red green strategy for meshing highly deformable objects with tetrahedra. In *International Meshing Roundtable*, vol. 12, 103–114.

NOORUDDIN, F. S., AND TURK, G. 2003. Simplification and repair of polygonal models using volumetric techniques. *IEEE Transactions on Visualization and Computer Graphics 9*, 2, 191–205.

O'ROURKE, J. 1998. *Computational Geometry in C*, 2nd ed. Cambridge University Press.

PERSSON, P.-O., AND STRANG, G. 2004. A simple mesh generator in matlab. *SIAM Review 46*, 2 (June), 329–345.

SETHIAN, J. A. 1999. Fast marching methods. *SIAM Rev. 41*, 2, 199–235.

SETHIAN, J. A. 1999. *Level Set Methods and Fast Marching Methods. Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge University Press. Cambridge Monograph on Applied and Computational Mathematics.

SIGG, C., PEIKERT, R., AND GROSS, M. 2003. Signed distance transform using graphics hardware. In *Proceedings of IEEE Visualization*, IEEE Computer Society Press, Seattle, WA, USA, 83–90.

SUD, A., OTADUY, M. A., AND MANOCHA, D. 2004. DiFi: Fast 3d distance field computation using graphics hardware. In *Proc. of Eurographics*, M.-P. Cani and M. Slater, Eds., vol. 23.

WDV'05 – WORKSHOP ON DYNAMICAL VISION AT ICCV'05, BEIJING, CHINA, OCTOBER 2005

A Batch Algorithm For Implicit Non-Rigid Shape and Motion Recovery

Adrien Bartoli
CNRS - LASMEA, France — Adrien.Bartoli@gmail.com

Søren I. Olsen
DIKU, Danemark — ingvor@diku.dk

Abstract

The recovery of 3D shape and camera motion for non-rigid scenes from single-camera video footage is a very important problem in computer vision. The low-rank shape model consists in regarding the deformations as linear combinations of basis shapes. Most algorithms for reconstructing the parameters of this model along with camera motion are based on three main steps. Given point tracks and the rank, or equivalently the number of basis shapes, they factorize a measurement matrix containing all point tracks, from which the camera motion and basis shapes are extracted and refined in a bundle adjustment manner. There are several issues that have not been addressed yet, among which, choosing the rank automatically and dealing with erroneous point tracks and missing data.

We introduce theoretical and practical contributions that address these issues. We propose an implicit imaging model for non-rigid scenes from which we derive non-rigid matching tensors and closure constraints. We give a non-rigid Structure-From-Motion algorithm based on computing matching tensors over subsequences, from which the implicit cameras are extrated. Each non-rigid matching tensor is computed, along with the rank of the subsequence, using a robust estimator incorporating a model selection criterion that detects erroneous image points.

Preliminary experimental results on real and simulated data show that our algorithm deals with challenging video sequences.

1. Introduction

Structure-From-Motion – the recovery of 3D shape and camera motion from images – is one of the most studied problems in computer vision. The decades of work has led to significant successes, especially when the observed environment is static. However, the assumption of rigidity is violated in many cases of interest, for example expressive faces, moving cars, etc. For that reason, dealing with non-rigid scenes coming from single-camera footage has received an increasing attention over the last few years. The problem is highly challenging since both the camera motion and the non-rigid 3D shape have to be recovered. A major step forwards for such cases was made by Bregler *et al.* [5, 9], Brand [4] and Aanæs *et al.* [1]. Building on the work of [2, 7], they developed and demonstrated factorization of images of non-rigid scenes, where the non-rigidity was represented as a linear combination of *basis shapes*. Xiao *et al.* [14] studied the degenerate deformations that may defeat the reconstruction algorithms.

This paper tackles the two following open problems. *(i)* the factorization of a measurement matrix containing all point tracks in the presence of missing and erroneous image points. This must be done to recover the parameters of the implicit imaging model. Most previous work do not deal with missing data [1, 4, 5, 9, 13]. *(ii)* the automatic choice of the rank r of the measurement matrix, characterising the degree of non-rigidity in the sequence. Most previous work rely on a user-defined rank [4, 5, 9, 10, 13].

More precisely, we build on the low-rank shape model to derive an *implicit imaging model* projecting points affinely from $\mathbb{R}^r$ – the implicit shape points – onto the images using *implicit camera matrices*. The rank r reflects the degree of non-rigidity of the model and is thus a very important parameter. This implicit model is simpler than the *explicit model* used in *e.g.* [5, 10], in the sense that it ignores the replicated block structure of the camera matrices. The implicit model gives weaker constraints on point tracks than the explicit model. It is the model used for non-rigid factorization in *e.g.* [5, 9, 13]. Based on this model, we derive *non-rigid matching tensors* that constrain point tracks and encapsulate information about the implicit camera matrices. We define non-rigid closure constraints relating the matching tensors to the implicit camera matrices. These theoretical concepts are based on the fact that implicit reconstruction is performed in $\mathbb{R}^r$. They lead to a batch algorithm for computing the motion and structure matrices in the presence of erroneous and missing data. The idea is to robustly compute a set of matching tensors over several subsequences using MAPSAC and the GRIC criterion to choose the associated rank [8]. From these matching tensors, we solve for the implicit camera matrices using the closure constraints. The next step consists in computing the basis shapes by non-rigid triangulation. We refine both the implicit cameras and implicit shape in a bundle adjustment manner. Finally, each image point is classified as an inlier or an outlier. Almost all steps in this algorithm are done robustly, meaning that blunders are detected and thus do not corrupt the computation.

Roadmap. In §2, we derive the non-rigid shape and imaging models. We examine previous work in §3. We derive the non-rigid matching tensors and closure constraints in §§4 and 5 respectively. Our Structure-From-Motion algorithm is derived in §6 while the robust estimation of matching tensors and associated ranks is given in §7. Experimental results are reported in §8 and our conclusions in §9.

Notation. Vectors are denoted using bold fonts, *e.g.* $\mathbf{x}$ and matrices using sans-serif or calligraphic characters, *e.g.* M or $\mathcal{X}$. Index $i = 1, \ldots, n$ is used for the images, $j = 1, \ldots, m$ for the points and $k = 1, \ldots, l$ for the basis shapes, *e.g.* $\mathbf{x}_{ij}$ is the position of the j-th point track in the i-th image and $\mathbf{B}_{kj}$ is the k-th basis shape for the j-th point. Visibility indicators modeling occlusions are denoted v_{ij}. The Hadamard (element-wise) product is written $\odot$. The zero and one vectors are respectively $\mathbf{0}$ and $\mathbf{1}$, 0 is the zero matrix and $^\mathsf{T}$ is vector and matrix transpose. Bars indicate centred data, as in *e.g.* $\bar{\mathcal{X}}$. Notation $[i, i']$ refers to a subsequence between image i and image i', *e.g.* $\mathcal{X}_{[i,i']}$ is the measurement matrix for this subsequence. $\{\}$ is a set over some variable. We use the Singular Value Decomposition, denoted SVD, *e.g.* $\mathcal{X} = \mathsf{U}\Sigma\mathsf{V}^\mathsf{T}$ where U and V are orthonormal matrices, and Σ is diagonal, containing the singular values of $\mathcal{X}$ in decreasing order.

Noise distribution. The noise on image point positions is supposed to be centred Gaussian i.i.d. Under this hypothesis, minimizing the $\mathcal{L}_2$-norm between measured and predicted point positions, often dubbed the reprojection error, yields Maximum Likelihood Estimates.

2. Non-Rigid Imaging Model

We review the low-rank shape model, dubbed the explicit model and derive our implicit model.

2.1. Explicit Model

The low-rank shape assumption consists in writing the coordinates of a time-varying set of points $\mathbf{Q}_{ij}$ as linear combinations over l *basis shapes* $\mathbf{B}_{kj}$ with the *configuration weights* α_{ik}: $\mathbf{Q}_{ij} = \sum_{k=1}^{l} \alpha_{ik}\mathbf{B}_{kj}$. Points $\mathbf{Q}_{ij}$ are projected onto the images by affine cameras: $\mathbf{x}_{ij} = \mathsf{P}_i\mathbf{Q}_{ij} + \mathbf{t}_i$, from which the explicit imaging model is obtained:

$$\mathbf{x}_{ij} \quad = \quad \mathsf{P}_i \left(\sum_{k=1}^{l} \alpha_{ik}\mathbf{B}_{kj} \right) + \mathbf{t}_i. \tag{1}$$

This trilinear equation is the most explicit form of the low-rank shape imaging model. Only rank-3 basis shapes are considered for simplicity, but rank-2 and rank-1 basis shapes can be modeled as well [14].

2.2. Implicit Model

Rewriting (1), one obtains:

$$\begin{aligned} \mathbf{x}_{ij} &= \begin{pmatrix} \alpha_{i1}\mathsf{P}_i & \cdots & \alpha_{il}\mathsf{P}_i \end{pmatrix} \begin{pmatrix} \mathbf{B}_{1j} \\ \vdots \\ \mathbf{B}_{lj} \end{pmatrix} + \mathbf{t}_i \\ &= \mathsf{M}_i\mathbf{S}_j + \mathbf{t}_i \qquad \text{with} \\ \mathsf{M}_i &= \begin{pmatrix} \alpha_{i1}\mathsf{P}_i & \cdots & \alpha_{il}\mathsf{P}_i \end{pmatrix}. \end{aligned} \tag{2}$$

We call M_i a $(2 \times 3l)$ *explicit camera matrix* and $\mathbf{S}_j^\mathsf{T} = \begin{pmatrix} \mathbf{B}_{1j}^\mathsf{T} & \cdots & \mathbf{B}_{lj}^\mathsf{T} \end{pmatrix}$ a $(3l \times 1)$ *shape vector*. Introduce $r = 3l$, the rank of the model, a $(r \times r)$ full-rank matrix $\mathcal{A}$ and relaxing the replicated structure yields the bilinear *implicit model*. From (2), $\mathbf{x}_{ij} = \mathsf{M}_i\mathbf{S}_j + \mathbf{t}_i = \left(\mathsf{M}_i\mathcal{A}^{-1}\right)\left(\mathcal{A}\mathbf{S}_j\right) + \mathbf{t}_i$, giving:

$$\mathbf{x}_{ij} \quad = \quad \mathsf{J}_i\mathbf{K}_j + \mathbf{t}_i. \tag{3}$$

We call $\mathsf{J}_i = \mathsf{M}_i\mathcal{A}^{-1}$ and $\mathbf{K}_j = \mathcal{A}\mathbf{S}_j$ the *implicit camera matrix* and the *implicit shape matrix* respectively. Matrix $\mathcal{A}$ represents a *corrective transformation*. As shown in the next section, this is the model used for non-rigid factorization. The model generalizes, in some sense, the $\mathbb{P}^k \rightarrow \mathbb{P}^2$ projection matrices introduced by Wolf *et al.* [12].

3. Previous Work

Most of the previous work [1, 4, 5, 9, 13] is based on factorizing a measurement matrix using SVD and hence do not cope with missing data. We note that Torresani *et al.* [10] propose an approach where the likelihood of the explicit model is maximized over the entire image sequence using a generalized EM (Expectation Maximization) algorithm which finds the nearest local optimum. The important rank selection problem is neglected in most papers, besides [1]. Below, we describe the three main steps involved in most algorithms. The inputs are the complete measurement matrix $\mathcal{X}$ and the rank r. The outputs are the camera pose, the configuration weights and the basis shapes.

Step 1: Factorizing. A $(2n \times m)$ measurement matrix $\mathcal{X}$ is built by gathering all point coordinates. The translation part of the imaging model, *i.e.* the $\mathbf{t}_i$, is estimated as the mean of the point coordinates in each image. A $(2n \times 1)$ joint translation vector $\mathbf{t}^\mathsf{T} = (\mathbf{t}_1^\mathsf{T} \ \cdots \ \mathbf{t}_n^\mathsf{T})$ is built and used to centre the measurement matrix: $\bar{\mathcal{X}} \leftarrow \mathcal{X} - \mathbf{t} \cdot \mathbf{1}^\mathsf{T}$, from which we get:

$$\underbrace{\begin{pmatrix} \mathbf{x}_{11} & \cdots & \mathbf{x}_{1m} \\ \vdots & \ddots & \vdots \\ \mathbf{x}_{n1} & \cdots & \mathbf{x}_{nm} \end{pmatrix}}_{\bar{\mathcal{X}}_{(2n \times m)}} = \underbrace{\begin{pmatrix} \mathsf{J}_1 \\ \vdots \\ \mathsf{J}_n \end{pmatrix}}_{\mathcal{J}_{(2n \times r)}} \underbrace{\begin{pmatrix} \mathbf{K}_1 & \cdots & \mathbf{K}_m \end{pmatrix}}_{\mathcal{K}_{(r \times m)}},$$

where $\mathcal{J}$ and $\mathcal{K}$ are the joint implicit camera and shape matrices. The centred measurement matrix is factorized using SVD as $\bar{\mathcal{X}} = \mathsf{U}\Sigma\mathsf{V}^\mathsf{T}$. The joint implicit camera and shape matrices $\mathcal{J}$ and $\mathcal{K}$, are recovered as the r leading columns of *e.g.* U and $\Sigma\mathsf{V}^\mathsf{T}$ respectively.

Step 2: Upgrading. The implicit model is upgraded to the explicit one by computing a corrective transformation. Xiao *et al.* [13] show that constraints on both the explicit camera and shape matrices must be considered to achieve a unique solution, namely the 'rotation' and the 'basis' constraints. They give a closed-form solution based on these constraints. Previous work [4, 5, 9] use only the rotation constraints, leading to ambiguous solutions. For instance, Brand [4] shows that a block-diagonal corrective transformation is a good practical approximation. Once the replicated structure has been approximately enforced, the rotation matrices are extracted using orthonormal decomposition. The configuration weights are then recovered using the orthonormality of the rotation matrices. Bregler *et al.* [5] assume that the information about each basis shape is distributed in the appropriate column triple in the shape matrix by the initial SVD, in other words that the entries off the block-diagonal of the corrective transformation matrix are negligible. Experiments show that this assumption restricts the cases that can be dealt with since only limited non-rigidity can be handled. A second factorization round on the reordered weighted motion matrix elements enforces the replicated block structure, yielding the weight factors and the P_i, which are upgraded to Euclidean by computing a linear transformation as in the rigid factorization case. Aanæs *et al.* [1] assume that the structure resulting from rigid factorization gives the mean non-rigid structure and camera motion. Given the camera motion, recovering the structure is done by examining the principal components of the estimated variance.

Step 3: Nonlinear refinement. The solution obtained so far is finely tuned in a bundle adjustment manner by minimizing *e.g.* the reprojection error. The algorithms proposed in [4, 9] differ by the prior they are using to regularize the solution. These priors state that the reconstructed shapes should not vary too much between consecutive images.

4. Non-Rigid Matching Tensors

Matching tensors are known for the rigid case. Examples are the fundamental matrix and the trifocal tensor. They relate the image position of corresponding points over multiple images. The implicit imaging model allows us to derive matching tensors for non-rigid scenes. These tensors are briefly mentioned in [6, §18.3.1].

A non-rigid matching tensor is a matrix $\mathcal{N}$ whose columns span the d dimensional nullspace of the $(2n \times m)$ centred measurement matrix $\bar{\mathcal{X}}$:

$$\mathcal{N}^\mathsf{T}\bar{\mathcal{X}} \;=\; 0. \tag{4}$$

The size of matrix $\mathcal{N}$ is $(2n\times d)$ where the tensor dimension is $d = 2n - r$. Loosely speaking, $\mathcal{N}$ constrain each point track $\bar{\mathbf{x}}_j$ – the j-th column of $\bar{\mathcal{X}}$ – by $\mathcal{N}^\mathsf{T}\bar{\mathbf{x}}_j = \mathbf{0}$. These constraints easily extend to the non centred measurement matrix $\mathcal{X}$ by substituting $\bar{\mathcal{X}} = \mathcal{X} - \mathbf{t}\cdot\mathbf{1}^\mathsf{T}$ into equation (4):

$$\begin{pmatrix}\mathcal{N}^\mathsf{T} & -\mathcal{N}^\mathsf{T}\mathbf{t}\end{pmatrix}\begin{pmatrix}\mathcal{X}\\ \mathbf{1}^\mathsf{T}\end{pmatrix} \;=\; 0.$$

Minimal number of points and views. The three following parameters are characteristic of an image sequence: the number of images n, the number of point tracks m and the rank r. They can be related to each other, in particular for, given r, deriving what the minimal number of point tracks and views are for computing the matching tensor. The computation is possible if the $(2n \times m)$ centred measurement matrix $\bar{\mathcal{X}}$ is at least of size $(r\times r)$. Counting the point track needed to compute the translations for centring the measurement matrix, we directly get the minimal number of point tracks as $m \geq r+1$. From $2n \geq r$, we obtain the minimal number of views as $n \geq \lfloor \frac{r}{2} \rfloor + 1$. These numbers can also be derived by counting the number of degrees of freedom in the tensor and the number of independent constraints given by equation (4).

Example: 2D rigid scene. In this case, $r = 2$ and pairs of points are related by a 2D affine transformation that can be estimated from 3 point correspondences. With centred coordinates, the relationship is $\bar{\mathbf{x}}_{2j} = \mathsf{A}\bar{\mathbf{x}}_{1j}$, *i.e.* :

$$\underbrace{\begin{pmatrix}\mathsf{A} & -\mathrm{I}\end{pmatrix}}_{\mathcal{N}^\mathsf{T}}\begin{pmatrix}\bar{\mathbf{x}}_{1j}\\ \bar{\mathbf{x}}_{2j}\end{pmatrix} \;=\; \mathbf{0},$$

from which we observe that the matching tensor has size (4×2). More generally, even-rank matching tensors predict an image point given all other $n-1$ image points.

Example: 3D rigid scene. In this case, $r = 3$ and pairs of points are related by the affine fundamental matrix that can estimated from 4 point correspondences. With centred coordinates, the relationship is $(\bar{\mathbf{x}}_2^\mathsf{T}\;\; 1)\bar{\mathsf{F}}_A(\bar{\mathbf{x}}_1^\mathsf{T}\;\; 1)^\mathsf{T} = 0$ with $\bar{\mathsf{F}}_A = \left(\begin{smallmatrix}0&0&a\\0&0&b\\c&d&0\end{smallmatrix}\right)$ the centred affine fundamental matrix:

$$\underbrace{\begin{pmatrix}c & d & a & b\end{pmatrix}}_{\mathcal{N}^\mathsf{T}}\begin{pmatrix}\bar{\mathbf{x}}_{1j}\\ \bar{\mathbf{x}}_{2j}\end{pmatrix} \;=\; \mathbf{0}.$$

More generally, odd-rank matching tensors predict the equivalent of an epipolar line in an image given all other $n-1$ image points.

OBJECTIVE

Given m point tracks over n images as a an incomplete $(2n \times m)$ measurement matrix $\mathcal{X}$ and a $(n \times m)$ visibility matrix $\mathcal{V}$, compute the implicit non-rigid cameras J_i, the non-rigid shape points $\mathbf{K}_j$ and the rank r.

ALGORITHM

1. Partition the sequence, see §6.1 while robustly computing the matching tensors $\{\mathcal{N}_{[i_b,i'_b]}\}$ and associated ranks, see §7.2.
2. Solve for the implicit cameras $(\mathsf{J}_i, \mathbf{t}_i)$ using the closure constraints, see §6.2.
3. Triangulate the point tracks to get the implicit shape points $\mathbf{K}_j$, see §6.3.
4. Nonlinearly refine the implicit cameras and shape points by minimizing the reprojection error, see §6.4.
5. Classify each image point track as an inlier or an outlier.

Table 1: Summary of our non-rigid implicit Structure-From-Motion algorithm.

5. Non-Rigid Closure Constraints

The closure constraints introducted by Triggs in [11] relate matching tensors to projection matrices. These constraints are used to derive a batch Structure-From-Motion algorithm dealing with high amounts of missing data.

In this section, we derive new types of closure constraints for the non-rigid case, based on the above-derived matching tensors, namely the $\mathcal{N}$-closure. Our derivation is valid for any rank r.

Let $\mathbf{K} \in \mathbb{R}^r$ be an implicit shape point. We project $\mathbf{K}$ in the images using the joint implicit camera matrix $\mathcal{J}$: $\bar{\mathbf{x}} = \mathcal{J}\mathbf{K}, \forall \mathbf{K} \in \mathbb{R}^r$. From the definition (4) of the matching tensors, $\mathcal{N}^\mathsf{T}\bar{\mathbf{x}} = \mathbf{0}$. Substituting the joint projection equation yields $\mathcal{N}^\mathsf{T}\mathcal{J}\mathbf{K} = \mathbf{0}, \forall \mathbf{K} \in \mathbb{R}^r$, which gives the *$\mathcal{N}$-closure constraint*:

$$\mathcal{N}^\mathsf{T}\mathcal{J} \quad = \quad 0. \tag{5}$$

This constraint means that the joint implicit camera matrix lies in the right nullspace of $\mathcal{N}^\mathsf{T}$.

6. Non-Rigid Structure-From-Motion

Our batch algorithm for implicit non-rigid Structure-From-Motion is based on the above-derived non-rigid matching tensors and closure constraints. It is summarized in table 1. We consider only sets of consecutive images for simplicity. It begins by selecting a set of s subsequences $\{[i_b, i'_b]\}_{b=1}^{b=s}$ and by computing a set of matching tensors $\{\mathcal{N}_{[i_b,i'_b]}\}$, one for each subsequence, and the associated rank estimates $\{r_{[i_b,i'_b]}\}$. Our joint tensor and rank estimation algorithm is presented in §7. The full sequence rank r is the maximum over all subsequence ranks: $r = \max_b(r_{[i_b,i_b]})$.

6.1. Partitioning the Sequence

The measurement matrix is partitioned into overlapping blocks with points visible in all of the selected images. Before going into further details, we must figure out what the minimal tensor dimension is, and how many views each tensor should operate on. Let $[i_b, i'_b]$ and $[i_{b+1}, i'_{b+1}]$ be two consecutive subsequences and let $\delta_{b,b+1} = i_{b+1} - i_b$ be the offset between them. We need to determine what the maximum value of $\delta_{b,b+1}$ is. The b-th matching tensor, with dimension $d_b = 2n_b - r_b$, gives d_b constraints. The number of unknowns constrained by the first matching tensor only is $\delta_{1,2}$, from which we get $\delta_{1,2} \leq n_1 - \lfloor \frac{r_1+1}{2} \rfloor$. Making the same reasoning for the b-th tensor, *i.e.* ignoring the constraints coming from previous overlapping sets, gives a bound on $\delta_{b,b+1}$:

$$\delta_{b,b+1} \quad \leq \quad n_b - \lfloor \frac{r_b + 1}{2} \rfloor. \tag{6}$$

Taking into account the other constraints lead to a tighter bound on $\delta_{b,b+1}$, but requires a cumbersome formalism to count the number of constraints and unknowns. Requiring $\delta_{b,b+1} > 0$ gives the minimal size of each image set as:

$$n_b \quad \geq \quad \lfloor \frac{r_b + 1}{2} \rfloor + 1. \tag{7}$$

For instance, for a 2D rigid scene, *i.e.* $r = 2$, the minimal n_b is 2 from equation (7) and the maximal $\delta_{b,b+1}$ is 1 from equation (6), *i.e.* using the affine transformations over pairs of consecutive views is fine. For a 3D rigid scene, *i.e.* $r = 3$, the minimal n_b is 3 and the maximal $\delta_{b,b+1}$ is 1, meaning that using trifocal tensors over triplets of consecutive of views is fine[1].

In practice, we do not know the ranks r_b at this step. We tune an initial guess while jointly partitioning the sequence and computing the matching tensors, as described in §7.2.

6.2. Solving For the Implicit Cameras

The leading part. We solve for the non-rigid cameras using the closure constraints. For each computed matching tensor, equation (5) gives the following constraints on the joint camera matrix $\mathcal{J}$:

$$\begin{pmatrix} 0_{(d_b \times 2(i_b - 1))} & \mathcal{N}^\mathsf{T}_{[i_b,i'_b]} & 0_{(d_b \times 2(n - i'_b))} \end{pmatrix} \mathcal{J} \quad = \quad 0.$$

Stacking the constraints for all $\{[i_b, i'_b]\}_{b=1}^{b=s}$ yields an homogeneous system $\mathsf{A}\mathcal{J} = 0$. It must be solved, *e.g.* in the least-squares sense, while ensuring that matrix $\mathcal{J}$ has full

[1]Triggs [11] states this result and shows the equivalence of using pairs of fundamental matrices over triplets of consecutive views.

column rank: $\min_{\mathcal{J}} \|\mathsf{A}\mathcal{J}\|^2$ s.t. $\det(\mathcal{J}) \neq 0$. We replace the full column rank constraint by a column orthonormality constraint, *i.e.* $\mathcal{J}^\mathsf{T}\mathcal{J} = \mathrm{I}_{(r\times r)}$. Note that the latter implies the former. This is done without loss of generality since for any full column rank joint camera matrix $\mathcal{J}$, there exist several coordinate transformations, say $\mathsf{G}_{(r\times r)}$, such that $\mathcal{J}\mathsf{G}$ is column orthonormal. One such a transformation is given by the QR decomposition of $\mathcal{J} = \mathcal{J}'\mathsf{G}^{-1}$. The transformed problem is solved by using the SVD $\mathsf{A} = \mathsf{U}\Sigma\mathsf{V}^\mathsf{T}$. Matrix $\mathcal{J}$ is given by the r last columns of V. Note that matrix A typically has a band-diagonal shape that one might exploit to efficiently compute its singular vectors, see *e.g.* [3].

The translations. The implicit imaging model (3) is $\mathbf{x}_{ij} = \mathsf{J}_i\mathbf{K}_j + \mathbf{t}_i$. By minimizing a least-squares error over all image points, the translations $\mathbf{t}_i$ in the joint translation vector $\mathbf{t}$, along with the basis shape vectors $\mathbf{K}_j$ can be reconstructed. We prefer to postpone the basis shape vector reconstruction to the next step, for robustness purposes. Instead, we consider the translation estimate $\mathbf{y}_{[i,i']}$ for each subsequence $[i,i']$, giving the centroid with respect to the points visible in the subsequence. We reconstruct these centroids along with vector $\mathbf{t}$. Note that in the absence of missing data, these centroids coincide. We minimize the reprojection error $\sum_{b=1}^{s} \|\mathbf{y}_{[i_b,i'_b]} - \mathcal{J}_{[i_b,i'_b]}\mathbf{Y}_{[i_b,i'_b]} - \mathbf{t}_{[i_b,i'_b]}\|^2$, where $\mathcal{J}_{[i,i']}$ and $\mathbf{t}_{[i,i']}$ are respectively a partial joint projection matrix and a partial joint translation vector restricted to the subsequence $[i,i']$, and $\mathbf{Y}_{[i,i']}$ is the reconstructed centroid. By expanding the cost function, the reprojection error is rewritten $\|\mathsf{A}\mathbf{w} - \mathbf{b}\|^2$, where the unknown vector $\mathbf{w}$ contains the $\mathbf{Y}_{[i_b,i'_b]}$ and $\mathbf{t}$. The solution is given by using the pseudo-inverse of matrix A, as $\mathbf{w} = \mathsf{A}^\dagger\mathbf{b}$. One must use a pseudo-inverse, since there is a r-dimensional ambiguity, making A rank deficient with a left nullspace of dimension r. This is a translational ambiguity between the basis shapes and the joint translation $\mathbf{t}$, that one can see by considering that $\forall\boldsymbol{\gamma} \in \mathbb{R}^r$, $\mathbf{x}_j = \mathcal{J}\mathbf{K}_j + \mathbf{t} = \mathcal{J}(\mathbf{K}_j - \boldsymbol{\gamma}) + \mathcal{J}\boldsymbol{\gamma} + \mathbf{t} = \mathcal{J}\mathbf{K}'_j + \mathbf{t}'$, with $\mathbf{K}'_j = \mathbf{K}_j - \boldsymbol{\gamma}$ and $\mathbf{t}' = \mathcal{J}\boldsymbol{\gamma} + \mathbf{t}$.

6.3. Reconstructing the Implicit Shape Points

We compute the basis shape vectors by non-rigid triangulation. This is done by minimizing the reprojection error. Assume that the j-th point is visible in the subsequence $[i,i']$, then this is formulated by:

$$\min_{\mathbf{K}_j} \|\bar{\mathbf{x}}_{[i,i']} - \mathcal{J}_{[i,i']}\mathbf{K}_j\|^2,$$

with $\bar{\mathbf{x}}_{[i,i']} = \mathbf{x}_{[i,i']} - \mathbf{t}_{[i,i']}$. The solution is $\mathbf{K}_j = \mathcal{J}^\dagger_{[i,i']}\bar{\mathbf{x}}_{[i,i']}$. We perform the minimization in a robust manner to eliminate erroneous image points. We use a RANSAC-like algorithm with adaptive number of trials. The number of image points sampled in the inner loop is $\lfloor \frac{r}{2} \rfloor + 1$.

6.4. Nonlinear Refinement

We complete the reconstruction algorithm by minimizing the reprojection error in order to finely tune the estimate:

$$\min_{\mathcal{J},\mathbf{t},\mathcal{K}} \|\mathcal{V}^+ \odot (\mathcal{X} - \mathcal{J}\mathcal{K} - \mathbf{t}\cdot\mathbf{1}^\mathsf{T})\|^2,$$

where $\mathcal{V}^+$ is obtained by duplicating[2] each row of the $(n \times m)$ visibility matrix $\mathcal{V}$. The minimization is done in a bundle adjustment manner. More precisly, we use a damped Gauss-Newton algorithm with a robust kernel. The damping is important to avoid singularities in the Hessian matrix, due to the $r(r+1)$ dimensional coordinate frame ambiguity. Contrarily to the explicit case, see [1, 13], no extra regularizing constraint is necessary.

7. Estimating the Non-Rigid Matching Tensors and Ranks

Our method estimates a non-rigid matching tensor over a (sub)sequence, *i.e.* for a complete measurement matrix, in a Maximum Likelihood framework. First, we tackle the case where the data do not contain outliers, and when the rank is given. Second, we examine the case where the data may contain outliers, and when the rank have to be estimated.

7.1. Outlier-Free Data, Known Rank

We describe a Maximum Likelihood Estimator, that handles minimal and redundent data. The translation $\mathbf{t}$ is obtained by averaging the point positions, and the measurement matrix is then centred as $\bar{\mathcal{X}} = \mathcal{X} - \mathbf{t}\cdot\mathbf{1}^\mathsf{T}$. The problem of finding the optimal $\mathcal{N}$ is formulated by $\min_{\hat{\mathcal{X}}} \|\bar{\mathcal{X}} - \hat{\mathcal{X}}\|^2$ s.t. $\mathcal{N}^\mathsf{T}\hat{\mathcal{X}} = 0$, where $\hat{\mathcal{X}}$ contains predicted point positions. This is a matrix approximation problem under rank deficiency constraint. It is solved by computing the SVD $\bar{\mathcal{X}} = \mathsf{U}\Sigma\mathsf{V}^\mathsf{T}$, from which $\hat{\mathcal{X}}$ is obtained by nullifying all but the r leading singular values in Σ and recomposing the SVD. Matrix $\mathcal{N}$ is given by the $2n - r$ last columns of U.

7.2. Contaminated Data, Unknown Rank

In most previous work, the rank of the sequence is assumed to be given. One exception is Aanæs *et al.* [1] who use the BIC model selection criterion to select the rank, but do not deal with blunders. When one uses subsequences, the subsequence rank may be lower than the sequence rank, and must be estimated along with the matching tensor. In addition, one has to deal with erroneous image points. We propose to use the robust estimator MAPSAC in conjunction with the GRIC model selection criterion proposed in [8]. GRIC is a modified BIC for robust least-squares problems. Our algorithm maximizes the GRIC score, as follows. In the inner

[2]This is simply to make it the same size as $\mathcal{X}$.

loop of the robust estimator, we sample point tracks and not only compute a single matching tensor, but multiple ones by varying the rank. Obviously, an upper bound r_{max} on the rank is necessary to fix the number of point tracks that one samples at each trial. One must take into account that the computational cost rises with r_{max}. One possible solution is to divide the sequence of trials into groups using gradually narrower intervals of possible rank values. The GRIC score is given by:

$$\text{GRIC} \quad = \quad \sum_{j=1}^{m} \rho\left(\frac{e_j^2}{\sigma^2}\right) + \lambda d + rm\log(m),$$

where e_j is the prediction error for the j-th point track, $\lambda = 4d\log(z) - \log(2\pi\sigma^2)$ and z is chosen as the image side length. Function ρ is $\rho(x) = x$ for $x < t$ and $\rho(x) = t$ otherwise, where the threshold $t = 2\log(\theta) + d\lambda/(2n)$ with θ the ratio of the percentage of inliers to the percentage of outliers. The noise level is robustly estimated using the weakest model, *i.e.* for a tensor dimension $d = 1$, as $\sigma^2 = \text{med}(e_j^2)/0.6745^2$. We refer the reader to [8] for more details.

8. Experimental Results

Most other methods do not handle missing data, and hence can not be compared to our. The method from Torresani *et al.* [10] handles missing data but uses the explicit model.

8.1. Simulated Data

We simulated $n = 180$ cameras observing a set of $m = 1000$ points generated from $l = 5$ basis shapes, hence with rank $\underline{r} = 3l = 15$. The configuration weights are chosen in order to give a decaying energy to successive deformation modes. The simulation setup produces a complete measurement matrix $\tilde{\mathcal{X}}$, from which we extract a sparse, band-diagonal measurement matrix $\mathcal{X}$, similar to what a real intensity-based point tracker would produce. A Gaussian centred noise with variance $\sigma^2 = 1$ is added to the image points.

In the experiments, we measured the *reprojection error* and the *generalization error*, which are dubbed in a machine learning context *training* and *test* error respectively. The reprojection error is $\mathcal{E} = \sqrt{\frac{1}{e}\|\mathcal{V}^{+} \odot (\mathcal{X} - \mathcal{J}\mathcal{K} - \mathbf{t}\cdot\mathbf{1}^{\mathsf{T}})\|^2}$, where e is the total number of visible image points. In other words, the reprojection error reflects the difference between the measures and the predictions. The generalization error is given by $\mathcal{G}_\gamma = \sqrt{\frac{1}{e_\gamma}\|\tilde{\mathcal{V}}_\gamma^{+} \odot (\tilde{\mathcal{X}} - \mathcal{J}\mathcal{K} - \mathbf{t}\cdot\mathbf{1}^{\mathsf{T}})\|^2}$, where γ indicates the percentage of hidden image points in $\tilde{\mathcal{X}}$ involved in the estimation and e_γ is the total number of image points used in the calculation. The $(n \times m)$ matrix $\tilde{\mathcal{V}}_\gamma$ indicates which image points are used in the calculation: it is constructed by including points further away

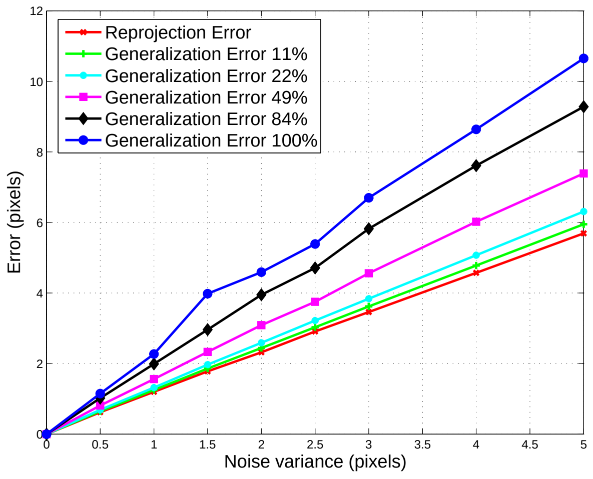


Figure 1: Reprojection and generalization error versus the variance of added noise σ for different percentages γ of hidden points to compute the generalization error.

from the visible points area while γ grows, *i.e.* $\tilde{\mathcal{V}}_0 = \mathcal{V}$ and $\tilde{\mathcal{V}}_{100} = 1_{(n\times m)}$. For example, $\mathcal{G}_0 = \mathcal{E}$ and $\mathcal{G}_{100} = \sqrt{\frac{1}{nm}\|\tilde{\mathcal{X}} - \mathcal{J}\mathcal{K} - \mathbf{t}\cdot\mathbf{1}^{\mathsf{T}}\|^2}$, *i.e.* all the visible and hidden image points are used to compute the error. Obviously, we expect the generalization error to be greater than the reprojection error, and to grow with γ.

The first experiment we performed consists in varying the level of added noise σ for different percentages γ of hidden points to compute the generalization error. The results are shown on figure 1. We observed that the reprojection error is slightly higher than the level of noise. The ability to generalize is accurate for a 1 pixel noise level, and smoothly degrades for larger noise levels, but is still reasonable: in the tested rang $\sigma = 0, \ldots, 5$ pixels, the $\gamma = 100\%$ generalization error is slightly higher than twice the noise level.

The second experiment we performed consists in varying the rank used in the computation, namely we tested $r = 11, \ldots, 27$, for different percentages γ of hidden points to compute the generalization error. The results are shown on figure 2. We observed that it is preferable to overestimate rather than to underestimate the rank, up to some upper limit. A similar experiment with roughly equal magnitude configuration weights to generate the data shows that r can be slightly underestimated and largely overestimated. The conclusion is that in practice, overestimating the rank is safe.

The third experiment is devised to assess the quality of the rank estimation based on GRIC in the presence of outliers. We tested for true ranks in the range $r = 3, \ldots, 18$ which covers what one expects to meet in practice. The results we obtained are shown in table 2, which shows av-

	3	6	9	12	15	18
0%	3.82	6.06	8.48	11.28	13.82	16.22
10%	3.86	6.02	8.60	11.02	13.66	16.24
20%	3.72	5.98	8.48	11.20	13.84	16.44
30%	3.64	5.94	8.52	11.00	13.52	16.58
40%	3.60	5.98	8.44	11.00	13.58	16.28
50%	3.40	5.88	8.30	10.86	13.68	16.16

	3	6	9	12	15	18
0%	0.38	0.42	0.57	0.66	0.65	1.12
10%	0.35	0.37	0.49	0.65	0.55	1.14
20%	0.45	0.37	0.50	0.60	0.58	0.50
30%	0.48	0.37	0.57	0.53	0.61	0.67
40%	0.49	0.32	0.57	0.53	0.64	1.08
50%	0.49	0.62	0.70	0.63	0.71	1.17

Table 2: (left) Average estimated rank r and (right) its standard deviation σ_r versus the true rank $\underline{r}$ and percentage of outliers.

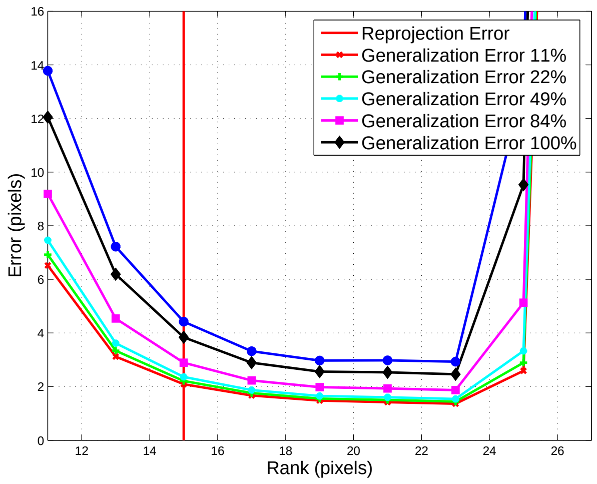


Figure 2: Reprojection and generalization error versus the rank r for different percentages γ of hidden points to compute the generalization error. The true rank $\underline{r} = 15$ is indicated with a vertical bar.

erages over 50 trials. We observed that these results are acceptable, even if the GRIC criterion we used is slightly biased since low ranks, *i.e.* less than 6, are slightly overestimated, while larger ranks, *i.e.* greater than 9 are slightly underestimated. It is however possible to correct for this bias in accordance with our conclusions on the previous experiment.

8.2. Real Data

We tested our algorithm on several image sequences. For one of them, extracted from the movie ‘Groundhog Day’, we show results. The sequence shows a man driving a car with a groundhog seated on his knees. The head of the man is rotating and deforming since he is speaking, and the animal is looking around, deforming its fur, opening and closing its mouth. Finally, the interior of the car is almost static, while the exterior is rigid, but moving with respect to the car.

The sequence contains 154 images, see figure 3 (top). We ran a KLT-like point tracker. We obtained a total of 1502 point tracks after having removed the small point tracks, namely which last less than 20 views. The visibility matrix, shown on figure 3 (bottom) is filled to 29.58%.

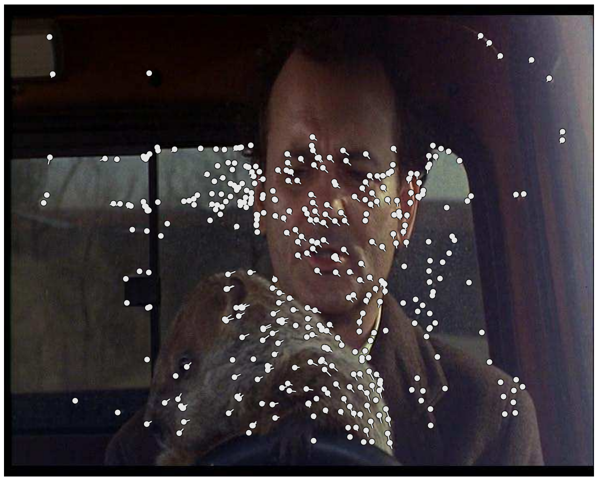


Figure 4: One frame with points and motion vectors reprojected from the reconstructed model.

For some parts of the sequence, where the motion of the different moving and deforming parts in the images is slow, computing the matching tensors is quite easy. Indeed, blunders can clearly be detected and classified as outliers. However, other parts in the sequence contain significant motion between single frames and motion blur occurs, making the point tracks slightly diverging from their ‘true’ position, and making the detection of outliers difficult. Large illumination changes sometimes make the tracker fails for entire areas of the image.

The reprojection errors we obtained at the non-rigid matching tensors estimation stage were distributed between 0.5 and 0.9 pixels, and 0.65 pixels on average. We used a user-defined rank $r = 15$. The initialization step yielded 58021 inliers over 68413 image points, *i.e.* the inlier rate was 84.8%, with a reprojection error of 1.19 pixels. The robust bundle adjustment yielded 61151 inliers, *i.e.* the inlier rate was 89.4%, with a reprojection error of 0.99 pixels. We

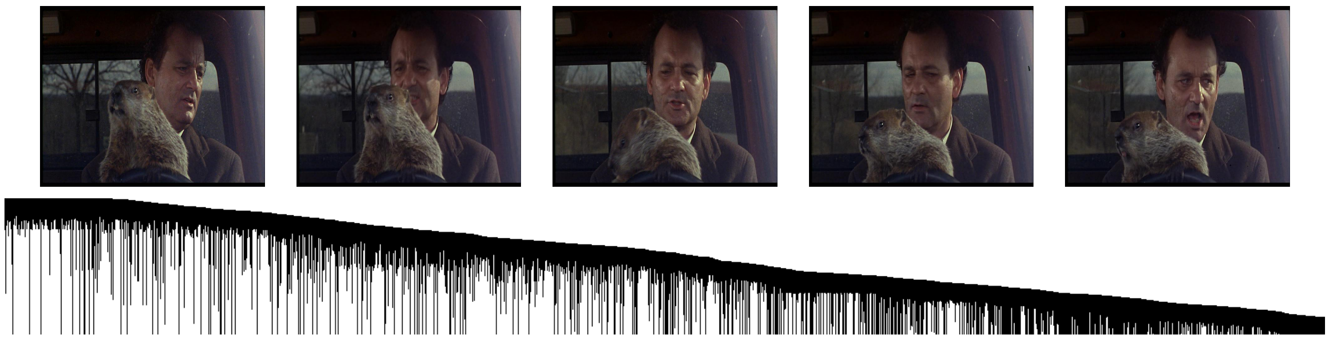


Figure 3: (top) 5 out of the 154 frames and (bottom) the visibility matrix $\mathcal{V}$ for the 'Groundhog Day' sequence.

believe it is a successful result on this challenging image sequence.

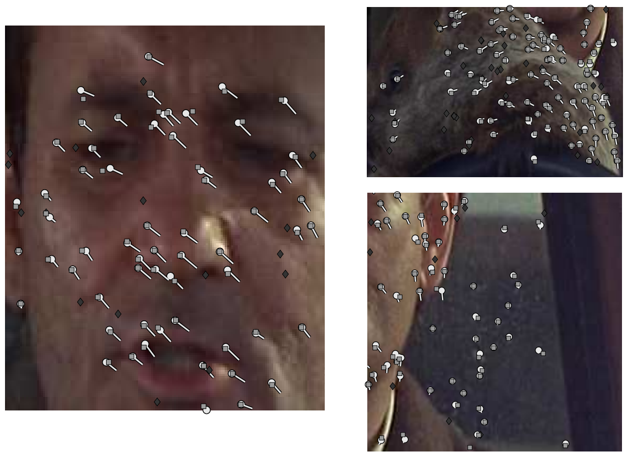


Figure 5: Closeup on the actor, the groundhog and the background overlaid with points and motion vectors reprojected from the reconstructed model (white dots), original points (light grey squares) and outliers (dark grey diamonds).

9. Conclusions

We proposed an implicit imaging model for non-rigid scenes, from which we derived non-rigid matching tensors and closure constraints. Based on these theoretical concepts, we proposed a robust batch implicit Structure-From-Motion algorithm for monocular image sequences of non-rigid scenes, dealing with missing data and blunders. Future work will be devoted to comparing various model selection criteria, and segmenting the scene based on the configuration weights, to recover objects that move or deform independently.

References

[1] H. Aanæs and F. Kahl. Estimation of deformable structure and motion. In *Proceedings of the Vision and Modelling of Dynamic Scenes Workshop*, 2002.

[2] B. Bascle and A. Blake. Separability of pose and expression in facial tracing and animation. In *Proceedings of the International Conference on Computer Vision*, 1998.

[3] A. Björck. *Numerical Methods For Least-Squares Problems*. Society For Industrial and Applied Mathematics, 1996.

[4] M. Brand. Morphable 3D models from video. In *Computer Vision and Pattern Recognition*, 2001.

[5] C. Bregler, A. Hertzmann, and H. Biermann. Recovering non-rigid 3D shape from image streams. In *Computer Vision and Pattern Recognition*, 2000.

[6] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.

[7] M. Irani. Multi-frame optical flow estimation using subspace constraints. In *Proceedings of the International Conference on Computer Vision*, 1999.

[8] P. H. S. Torr. Bayesian model estimation and selection for epipolar geometry and generic manifold fitting. *International Journal of Computer Vision*, 50(1):27–45, 2002.

[9] L. Torresani and C. Bregler. Space-time tracking. In *Proceedings of the European Conf. on Computer Vision*, 2002.

[10] L. Torresani and A. Hertzmann. Automatic non-rigid 3D modeling from video. In *Proceedings of the European Conference on Computer Vision*, 2004.

[11] B. Triggs. Linear projective reconstruction from matching tensors. *Image and Vision Computing*, 15(8), 1997.

[12] L. Wolf and A. Shashua. On projection matrices $P^k \rightarrow P^2$, $k = 3, \ldots, 6$, and their applications in computer vision. *International Journal of Computer Vision*, 48(1), June 2002.

[13] J. Xiao, J.-X. Chai, and T. Kanade. A closed-form solution to non-rigid shape and motion recovery. In *Proceedings of the European Conference on Computer Vision*, 2004.

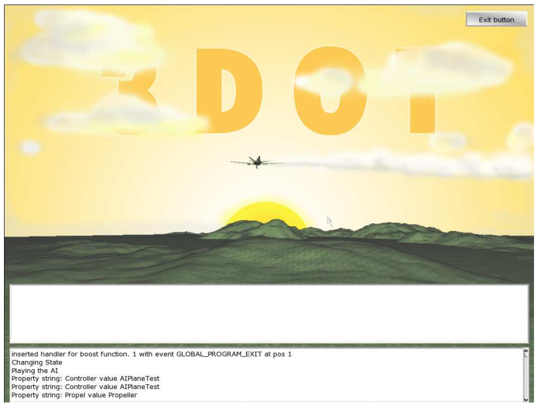
[14] J. Xiao and T. Kanade. Non-rigid shape and motion recovery: Degenerate deformations. In *International Conference on Computer Vision and Pattern Recognition*, 2004.

3DOT

An Open Source Game Engine

André Tischer

July 12, 2006



```
inserted handler for boost function. 1 with event GLOBAL_PROGRAM_EXIT at pos 1
Changing State
Playing the AI
Property string: Controller value AIPlaneTest
Property string: Controller value AIPlaneTest
Property string: Propel value Propeller
```

1 Abstract

This is a short description of the 3DOT game engine and its overall design. It is not meant as a complete guide but more as an introduction to 3DOT and large scale game engine design. It demonstrates the framework that controls the visual rendering process and how it can be altered during execution. We will not be diving into implementational details.

Note: this article is an extraction of the master thesis by André Tischer and what is presented here is only meant as a quick glance at game engine design and what it involves, for more detail see [Tis06]. High resolution versions of the figures can also obtained from that location.

2 3DOT - Game Engine

3DOT is a open source 3D game engine developed by students at the Department of computer science at University of Copenhagen. The engine is a large collection of different modules, which in many cases can be used as a stand alone program, i.e. out of the 3DOT context. The engine is designed as a multi-game environment. This means that 3DOT is not restricted to support any specific game or game type. The typical game made with 3DOT would be a real-time 3D game.

Some design rules have been enforced during the development of the 3DOT engine: the language that we have used to write comments and documentation is English. All modules are implementations of some 3DOT API, this is done in order to increase modularity of the engine. This also applies to all third party software and especially operating system specific modules. The APIs makes it possible to remove any part of the engine, so it becomes possible for the programmer to replace it with his own module, or improved tool, without having to rewrite the whole engine.

Further to ensure modularity the engine, including all engine modules, is kept free of any game application specific code. When creating a module that relate to a specific project, it can be added to the repository. The module is not considered a part of the basic functionality of the engine. This way it can be used by others and enrich all projects using the 3DOT engine.

3DOT closely incorporates a XML serialization system and a script language. This is a standardization of how to store and retrieve user data and it also makes it easy for the programmers to develop application specific and dynamic code using the script language. It promotes a fast way of prototyping and developing modules. XML has been chosen because:

- It is a very easy and standard method to create structured data
- Its wide spread use in many fields
- It is understood by many people, i.e. they do not have to learn some new file for-

mat in order to understand 3DOTs data files.

A simple gauge of XMLs popularity can be achieved with a search with the keyword XML using `http://www.google.com/`. This search yields over 1.9 billion hits within many different areas of subject.

Commercial game engines has a lot of runtime requirements, but as 3DOT is not developed with commercial purposes it does not have to meet the normal requirements of commercial game engines. This means that 3DOT can avoid the confining constraints that commercial engines has, such as the backwards compatibility, restrictions on memory and CPU, and diversity among specialized hardware. It is also hoped that not having these thigh runtime requirements, that the 3DOT engine can be used in research of future technology. 3DOT expects that the machine its running on is a relatively new machine in terms of memory, CPU, and graphics hardware. At the time of writing (summer 2006) the minimum requirements is: a 1.8 gHz CPU, 1 Gb of RAM, and a graphics card that is directX 9.0 compliant.

This section introduces the game engine, its overall design, and a walk through of how the modules fit's together when used as a complete solution.

2.1 General Design

The mantra of 3DOT development is modularity where possible. Figure 1 illustrates the main modules and how they relate. In the following the figure is examined in detail. Please refer to this figure during the explanation of the different modules.

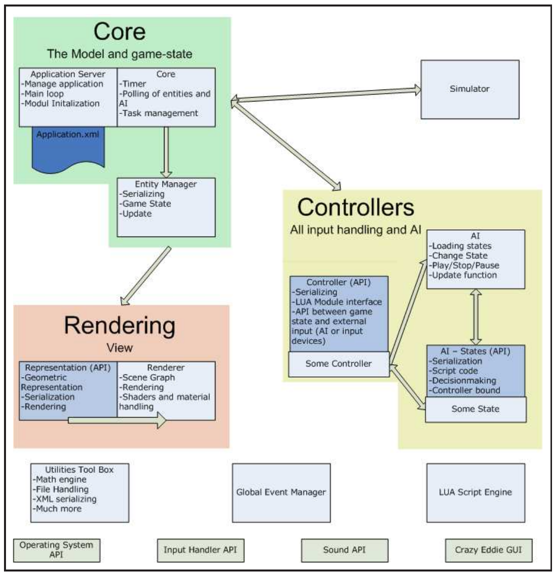


Figure 1: *Main design of 3DOT. It shows how the rendering, representation, and the control logic is separated into different modules.*

A main design idea we have adopted is the model-view-controller scheme [GHJV94]. The model-view-controller scheme "is a software architecture that separates an application's data model, user interface, and control logic into three distinct components so that modifications to one component can be made with minimal impact to the others." [Wik06b]. View means the presentation, i.e. the rendering in this case. This scheme applied to a game engine is described in [Rou05]. A nice consequence of this is for example the possibility of simultaneous development on the renderer and the AI system. It also means that we can shut off the rendering and still run the game should we wish to do that. This can be very useful if one for example want to implement a server for net-gaming.

2.1.1 Game State - Model

Below the tree main modules of the game state is explained:

Different games are called applications in 3DOT terminology. This is due to the `Application Server` module. This module is the C++ main function. What it does is to serialize an `XML` file that describes which modules to load, how to initialize them, and describes what to do in the render and main loop, also known as the game loop. This constellation makes it possible to have several applications running on the same engine. This is achieved simply by changing the `XML` application file and reloading the engine. The application makes sure to load the scene description as specified in the application file. The application server makes sure to initialize and start up the application.

The `Core` module is basically a process handler. It is the main scheduler in the engine just like the process task handler of an operating system. The scheduler is a singleton priority queue of function handles. Modules that must be updated at some frequency can register their update function in the scheduler along with the update time, i.e. the time until next call. The scheduling is implemented using voluntary process handling, this means that it is up to each function to yield or return before the function monopolizes the system. The reason that the scheduler uses this scheme, instead of automatic process handling, is because a lot of modules are dependant upon the same data. This data dependency demands that a module finishes its update before starting on another module. With automatic scheduling the developers have no control over which modules are in the process of being updated or not. Another issue for game developers is that the different modules have some guarantied update frequencies, like each frame, twenty times a second, and so on. Using a voluntary process handling makes it easier to achieve this. It can be imagined that some of the modules could be handled by a automatic scheduler, the advantages of having two schedulers has not been explored.

`Entities` are *things* within the game that have some kind of dynamic state data. An in-game character will always be an entity. It will as a minimum have a world position and orientation, but most likely also some internal memory (state), and current task. An entity could also be an invisible trigger that set off some alarm when some event occurs. However, an entity is not some static landscape or house placed in the game world. Entities are only things with some state data that changes, i.e. data that needs to be saved in order to replicate the situation correctly. An example of what is not an entity is the cloud system of the game Far Cry (Crytek 2004). It is not possible for the player to interact with the clouds and the positions of the individual clouds are not important for the game play in any other way than as a visual pleaser. Even though the clouds are animated, i.e. they have changing position, they are not considered entities in 3DOT terminology because of their lack of any state data.

The only place where game state data is stored is within the collection of entities. A consequence of this is that all other modules that requires access to game state data needs to interface with this module. Only two modules are allowed to modify the game state and that is either the simulator, which take care of natural motion, and controllers. Of cause the entities them selves can impose changes.

2.1.2 Controller API - Controller

The controller interface is the API that determines how to access a specific entity and the state data stored there. An entity can have many different controllers depending on the input device, AI, Human Interface Device (**HID**), and such. Most likely an entity will have a small tree of controllers, as figure 2 demonstrates. It is, however, not all entities that has a controller attached. It is only those that need input from outside the system. Input is defined as all external forces that cannot be considered a part of the model, for example gravity would not be an external force but adding a sudden force to an object in some direction.

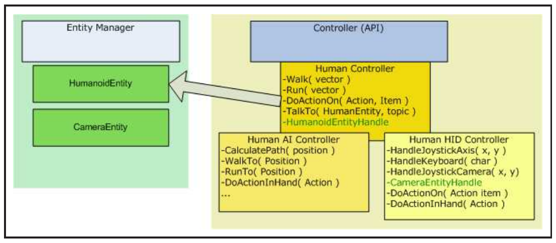


Figure 2: *Demonstration of how an entity can have more than one controller. Also notice that the child controllers implement different versions of the parent functions, depending on the input type.*

With this scheme we are able to make a super class that defines some common functionality and some requirements (C++'s pure abstract class) of all inheriting objects. This way it is possible to provide a uniform interface to each child controller, no matter whether they are AI or a HID. In many cases all types of input providers (AI or HID) will utilize the same interface.

Here the strength of the model-view-controller design scheme becomes apparent: the entity does not care about whether it is an AI or a HID that provides the input, as long as it uses the right mapping from whatever function the controller exposes to the entity functionality. This also provides some degree of freedom parallel development for designing new in-game items.

It is the responsibility of the controllers to translate the many different types of input into a uniform batch that can be applied to the entity. It is often the case that the controller is exposed to LUA so that it becomes easier to alter and fine tune the individual controller.

2.1.3 Simulator - Model

The main job of a simulator is to make objects move. Often it simulates an earth like environment and objects in the world (entities) fall down when pushed over the edge. It is the job of the simulator to move all entities according to the physics settings of the particular game. It is also the job of the simulator to make sure that objects do not intersect with each other - collision detection. The simulator is the only module that is voluntary. Many games, such as typical RTS games, do not have a global simulator. The main reasons to skip this module is to save development time, to minimize the amount of risk involved in the development process, to gain performance, and the fact that game designers are just warming up to the challenge of using simulated physics as an integrate part of the game design. The many tasks that the simulator performs can be faked to some extent. For example, walking on the terrain may be done by constraining all movement to the 2d ground plane, determining the height over this plane by applying the landscape height at this position, and by playing some prerecorded animation at some appropriate pitch. To avoid collision detection the path planner can simply make sure that it avoids all static objects. If a few objects need to fall then just implement a very simple simulator within the entity AI program. This method is used and is a very effective way of controlling the computer resources, but more and more games incorporates a full simulation system.

A simulator is regarded as a part of the model. Even though the simulator is modifying the game state, a simulator is not a module that is closely controlled and it more or less takes care of itself as long as it gets CPU time. Another reason to regard the simulator as a part of the model is the close relationship between it self and the entities. All entities must be developed in a way such that they expose the information needed by the simulator and they do not have to update their own data. Note that not all entities have to be simulated. It is often the case that only a few objects are simulated, and once an object becomes static again it is put to *sleep* and removed from the list of simulated objects. Sleeping objects are all objects that are not moving or has so little movement energy that they are clamped to be static. The sleeping/simulated object method is used in order to save processor time.

3DOT is designed to interface with a broad range of simulators. It is a non-trivial thing to implement a simulator and to integrate it into an engine. This process must be done with care. Entities that are moved by the simulator must still have all data within the entity while working with the simulator. This can be achieved by a double representation or by making the simulator acquire the data directly from the entities.

The 3DOT engine has been working with two different simulators over the years, the RenderLoop engine and a simulator provided from OpenTissue [EDSH01]. The example game of this thesis has no simulator attached. All motion projection is done by the entities themselves.

2.1.4 Renderer - View

3DOT uses a double representation of all data used in the rendering process. This means, that the renderer has its own transformation data even though it is the entities that ultimately decide where all game objects are located. This double representation has been chosen for two reasons: Not all game objects that has to be rendered is entities, for example the landscape and clouds from previous examples. The renderer has its own scene graph it stores all needed information in and utilizes it at each render event

A scene graph can be many things, but normally as a minimum, the term covers a transformation hierarchy, i.e. a tree structure where each node is a geometric entity with a transformation and a orientation that transforms into its parent object space. The root node is the origin of the world space: $[0, 0, 0]$. This is a very convenient way of storing data when rendering the data and manipulating the data [WP00, chapter 9]. But from a simulator standpoint it might not be, for example an aero plane entity may consist of eight to ten different geometric nodes, wings, guns, fuselage, flaps, and so on, but it is only the root node of this sub tree that is needed when simulating. The rest of the nodes are expected to just follow their parent node. Also simulators work only in world space. It is not given that a geometric node in the scene graph necessarily is expressed in terms of world space, for example the flaps of the plane. Figure 3 shows both how a collection of geometric nodes can comprise one single entity, how the node tree (scene graph) looks, how an entity implementation can manipulate the geometric nodes, and how a controller and simulator can influence the system.

2.1.5 Utilities and APIs

Accompanying the central part of the game engine is an abundance of utility libraries. Most of these modules are more loosely connected to the engine than the modules described above. Here is a short list and descriptions of the central modules:

- **Math:** No game engine without an extensive math module. 3DOTs math module has been developed since before the start of 3DOT and is a very well tested module. The math module contains a broad range of types. With regard to game development the most basic are: quaternion, vectors, small and large matrices, and transformation library.
- **Audio:** The audio module describes a general API to a audio system. As of summer 2006 only a windows implementation of the API has been added. It uses `OpenAL` [ope06] as a basis. It is possible to create emitters and place them in the 3D game world and them attach sources to these emitters. Each emitter has a position, a orientation, and velocity. A source can either be streamed or immediate. Streamed sounds are handled as a separate thread whereas the immediate sound is loaded into memory and played directly from there.
- **Input Handler:** Like the audio module, and in accordance with the 3DOT design mantra, the input handler describes a generalized API for handling input from joystick, keyboard, mouse, and other types of HIDs - All special input devices such as dance mats, guitar controllers, drum controllers, registers themselves as joysticks. Only a `DirectInput` Microsoft implementation has been added at time of writing [dir05].
- **Operating System:** The operating system module is not quite a module in the standard way. No application without an interface to the operating system. This module wraps all functionality that goes to opening a window, and handling messages from the underlying operating system. Again, only a windows implementation has been implemented at the time of writing.

  Porting 3DOT to another operating system is in large part done by implementing this module. But some other modules must also be ported. All input and output modules must be able to run on the target system, audio, file IO, and such.
- **File IO:** Handling IO operations with regard to fetching of files from data storage, calculation of the correct paths and such. This module is operating system, file system, dependent. As all file handling is done through this module it is only this module that needs to be implemented in a new version if the engine is to be ported.
- **Crazy Eddie Graphics User Interface (GUI):** 3DOT incorporates a full third part open source GUI system. Crazy Eddie (CE) is a game oriented GUI subsystem [Wik06a]. It takes care of drawing all head-up-display (HUD) objects, such as dialog boxes, menus, sub windows, text input, and such. The Crazy Eddie homepage states that it is: ”targeted at games developers who should be spending their time creating great games, not building GUI sub-systems!” [Wik06a].
- **XML serialization:** Serialization to and from XML documents [xml06]. The way this module works is by forcing all serialize objects to implement a function that takes care of loading or saving, depending on the current activity. It is possible to call other serialize objects and their respective serialize function is then recursively called. This way it is possible to create a whole hierarchy of serialize functions.

- **Script Language - LUA:** 3DOT has chosen LUA script [lua06]. Scripting has become a very integrate part of many modern game engines. In 3DOT it has been chosen not to wrap an API around the script language. Due to compile time issues, we have chosen to gather all scripting in a few libraries, but as scripting is heavily used in many different situations, LUA library code can be found in many modules, but where possible it has been gathered in one library.

  The LUA programming language is not a new language it has been around since 1993. It is made by TeCGraf - Computer Graphics Technology Group at Pontifical Catholic University of Rio de Janeiro. It has over the years been used in several commercial products among these games such as World of Warcraft (Blizzard Entertainment 2005), Grim Fandango (Lucas Arts 1998), and Escape from Monkey Island (Lucas Arts 2000). Because of the very large community and the continued development of the language 3DOT has chosen to incorporate LUA closely into the engine. Together with `luabind`, a library that helps the creation of bindings between C++ and LUA, 3DOTs script engine is utilized in many modules and works well with the serialization system.

  For more information about LUA and `luabind` see [IdFC06] and [WN03]. The background of LUA is explained in [IdFC01].

- **Reflection System:** 3DOT has an automated reflection system that makes sure that some class, LUA or C++, can be automatically instantiated and that it can expose properties to the GUI system. This automates the process of making GUI components to each type and instance. When creating level altering tooles this module is the corner stone. It makes it possible to create all entities and GUI components dynamically.

- **Event Handler:** It can be very important to have a way of communicating between two arbitrary points within the game engine. The global event handler takes care of this functionality. From any where an object can register a event, just a string value for example ”GLOBAL_PROGRAM_EXIT”, and a handler callback function. When some other part of the program fires the event of that name, all waiting callback functions are called. It is also possible to parse data values via this interface.

  It shall however not be mistaken with the local event system that each entity can have attached. This system has the same basic functionality but the domain is only within the collection of entities. These local event managers are mostly used to build the game mechanics: when this lever is pulled the trap door opens.

Not all modules are listed here. Among the unlisted modules are the timer module, log file module, string utility, and more.

3 The Future

The 3DIKU engine is an engine in development. It is by far not a finished product. Before it can support any real large scale game development it needs an editor to support having non-tech people to create the levels and game worlds, more finish with regard to the whole use of the code base, such as a consistent documentation, i.e. tutorials, introduction to module design, overall design philosophy, and a function reference document. As 3DIKU have been a student driven project based upon the work done in a couple of master thesis', some written projects, and voluntary work, it has somewhat lacked behind with regard to Quality Assurance (**QA**) and documentation. These issues have however over first half of 2006 become a more important issue and steps to remedy these have been taken - this work being one of them. Also the people behind the 3DIKU engine have begun a collaboration with the people behind OpenTissue [EDSH01]. The aim is to provide 3DIKU with state of the art simulators to be used along side the 3DIKU engine. From the OpenTissue perspective the need for a advanced visualization, complete with GUI, input handling, scene handling and such, has been a long standing issue, as these areas are not within the scope of the OpenTissue charter.

Any questions can be directed to André Tischer at `tischer@diku.dk`.

References

[dir05] Direct input tutorials, 2005. `http://msdn.microsoft.com/library/default.asp?url=/library/en-us/directx9_c/dx9_directinput_tutorials.asp`, This is the place to start when whanting an introduction to programming HIDs under windows.

[EDSH01] Kenny Erleben, Henrik Dohlmann, Jon Sporring, and Knud Henriksen. Open tissue - middleware physics for games and surgical simulation, 2001. `http://www.opentissue.dk/`, OpenTissue is a low level application programming interface (API) and a playground for future technologies in middleware physics for games and surgical simulation.

[GHJV94] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns - Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1994.

[IdFC01] Roberto Ierusalimschy, Luiz Henrique de Figueiredo, and Waldemar Celes. The evolution of an extension language: A history of lua. Technical report, Proceedings of V Brazilian Symposium on Programming Languages B-14–B-28, 2001. `http://www.lua.org/history.html`, A detailed description of the early development of Sol wich eventually turned in to LUA.

[IdFC06] Roberto Ierusalimschy, Luiz Henrique de Figueiredo, and Waldemar Celes. Reference manual for lua 5.1. Technical report, Pontifical Catholic University of Rio de Janeiro, June 5 2006. `http://www.lua.org/manual/5.1/`, The survival guide to LUA. An up-to-date reference manual of LUA 5.1.

[lua06] Lua - the programmin language, 2006. `http://www.lua.org/`.

[ope06] Open al - cross-platform 3d audio, 2006. `http://www.openal.org/`, This libray is licensed under L-GPL and is a very used sound library.

[Rou05] Jorrit Rouw. The guerrilla guide to game code. *Gamasutra*, April 2005. `http://www.gamasutra.com/features/20050414/rouwe_01.shtml`, This is very fine article that describes a solid framework when designing a computer game engine.

[Tis06] André Tischer. André tischers home page, August 2006. `http://www.tischer/`.

[Wik06a] Crazy Eddie Wiki. Welcome to crazy eddie's gui system, June 2006. `http://www.cegui.org.uk/wiki/index.php/Main_Page`, This is the place to start when looking for more knowledge about Crazy Eddie GUI system.

[Wik06b] Wikipedia. Model-view-controller, June 2006. `http://en.wikipedia.org/wiki/Model-View-Controller`.

[WN03] Daniel Wallin and Arvid Norberg. Documentation of luabind. Technical report, Rasterbar Software - Software development and consulting, Januar 11th 2003. `http://www.rasterbar.com/products/luabind/docs.html`, This is a nice add on to LUA, but know about LUA before starting with luabind.

[WP00] Alan Watt and Fabio Policarpo. *3D Games: Real-Time Rendering and Software Technology, Volume 1*. Addison and Welsey, 2000.

[xml06] Extensible markup language (xml), 2006. `http://www.w3.org/XML/`.

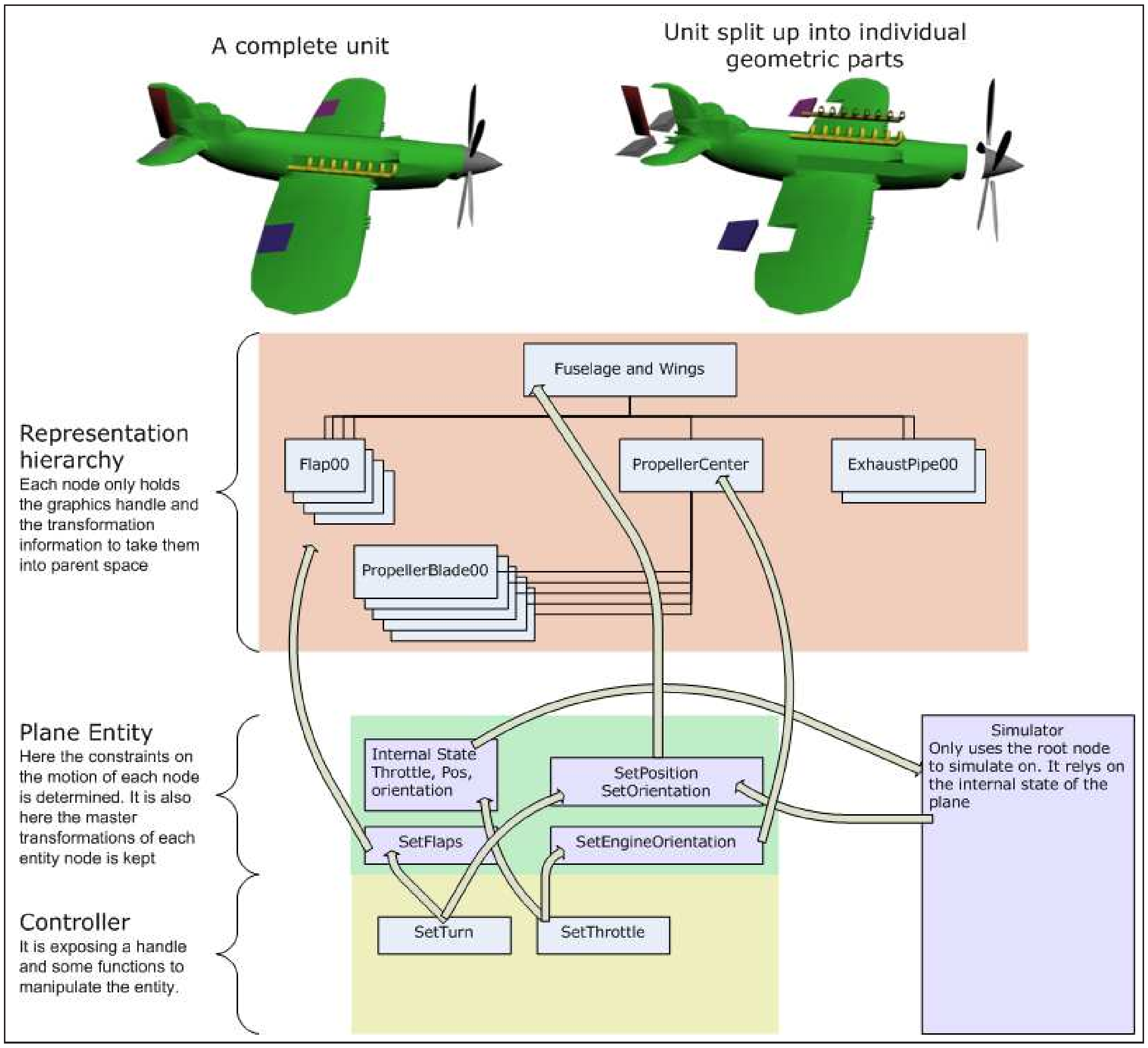

A complete unit
Unit split up into individual geometric parts
Fuselage and Wings
Representation hierarchy
Each node only holds the graphics handle and the transformation information to take them into parent space
Flap00
PropellerCenter
ExhaustPipe00
PropellerBlade00
Plane Entity
Here the constraints on the motion of each node is determined. It is also here the master transformations of each entity node is kept
Internal State Throttle, Pos, orientation
SetPosition SetOrientation
Simulator
Only uses the root node to simulate on. It relys on the internal state of the plane
SetFlaps
SetEngineOrientation
Controller
It is exposing a handle and some functions to manipulate the entity.
SetTurn
SetThrottle

Figure 3: *An overview of how the model-view-controller scheme looks in the 3DOT engine. The color scheme is adopted from figure 1. It shows the break-up of a geometric model, how it is arranged in a transformation hierarchy (scene graph), how the entity moves the different nodes around, and how the controller exposes functionality to induce changes into the system. The curved arrows indicate how influence is propagated around in the system. The plane model is created by Lars Westergaard Thomsen (2006) in 3D Studio Max.*

Towards Describing Crouzon Syndrome via a Craniofacial Atlas

Hildur Ólafsdóttir[1], Tron Darvann[1,2], Estanislao Oubel[3], Alejandro F. Frangi[3], Nuno V. Hermann[2,4], Bjarne K. Ersbøll[1], and Chad A. Perlyn[5]

[1] Informatics and Mathematical Modelling, Technical University of Denmark, ho@imm.dtu.dk

[2] 3D-Laboratory, School of Dentistry, University of Copenhagen; Copenhagen University Hospital

[3] Computational Imaging Lab, Department of Technology - D.326, Pompeu Fabra University, Barcelona

[4] Department of Pediatric Dentistry and Clinical Genetics, School of Dentistry, Faculty of Health Sciences, University of Copenhagen

[5] Division of Plastic Surgery, Washington University School of Medicine

1 Introduction

Crouzon syndrome was first described nearly a century ago when calvarial deformities, facial anomalies, and abnormal protrusion of the eyeball were reported in a mother and her son [1]. Later, the condition was characterized as a constellation of premature fusion of the cranial sutures (craniosynostosis), orbital deformity, maxillary hypoplasia, beaked nose, overcrowding of teeth, and high arched or cleft palate. Identification of heterozygous mutations in the gene encoding *fibroblast growth factor receptor type 2* (FGFR2) have been found responsible for Crouzon syndrome [2]. Recently a mouse model was created to study one of these mutations (FGFR2$^{\mathrm{Cys342Tyr}}$)[3]. This model allows for detailed examination of the craniofacial growth disturbances. The goal of this study is to automatically assess, visualise and statistically analyse these deviations in a set of adult wild-type (normal) mice and mice with Crouzon syndrome. This paper presents the preliminary steps towards these goals. Firstly, the construction of a nonrigid craniofacial wild-type (WT) mouse atlas. Secondly, the estimation of deformation fields from the atlas to all subjects using nonrigid registration.

The outline of the paper is the following. In the next section, data acquisition and methodology will be discussed. Section 3 presents the experimental results in terms of qualitative and quantitative, landmark-based registration accuracy. Section 4 provides a discussion of the results and conclusions.

2 Methods and Materials

2.1 Data

Production of the FGFR2$^{C342Y/+}$ and FGFR2$^{C342Y/C342Y}$ mutant mouse (Crouzon mouse) has been previously described [3]. All procedures were carried out in agreement with the United Kingdom Animals (Scientific Procedures) Act, guidelines of the Home Office, and regulations of the University of Oxford. Mutant mice of breeding age were determined by phenotype.

For three-dimensional (3D) CT scanning, 10 WT and 10 FGFR2$^{C342Y/+}$ specimens at six weeks of age (42 days) were sacrificed using Schedule I methods and fixed in 95% ethanol. They were sealed in conical tubes and shipped to the Micro CT imaging facility at the University of Utah. Images of the skull were obtained at approximately 46μm $\times$ 46μm $\times$ 46μm resolution using a General Electric Medical Systems EVS-RS9 Micro CT scanner. Figure 1 shows an example of the mice and imaging data appearance.

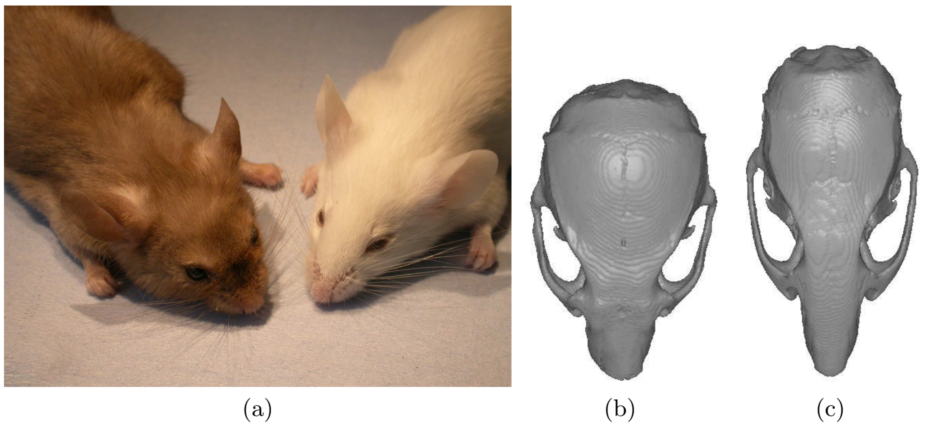


Fig. 1. (a) Photo of a Crouzon mouse (left) and a WT mouse (right). Skull surfaces extracted from CT images of (b) a Crouzon mouse, (c) WT mouse.

2.2 Nonrigid registration

The goal of image registration is to warp one image into the coordinate system of another using an optimal transformation $\mathbf{T}(x, y, z) \mapsto (x', y', z')$. A basic image registration algorithm requires the following:

- A transformation type.
- A measure of image similarity.
- An optimisation method to optimise the transformation parameters with respect to the similarity measure.

In this study the FFD-based nonrigid registration algorithm presented in [4] was adopted. In this approach, the transformation consists of both a global and a local model, i.e.

$$\mathbf{T}(x, y, z) = \mathbf{T}_{global}(x, y, z) + \mathbf{T}_{local}(x, y, z). \qquad (1)$$

The global transformation model describes the overall difference between the two images. This is achieved by an affine transformation. In this study, the affine transformation was defined by rotation and translation in addition to anisotropic scaling since the largest changes between the two groups stem from the differences in aspect ratio. This gives an affine transformation with 9 degrees of freedom. In this way, the largest part of the differences between the two groups of mice is covered by the affine registration.

However, local differences between the groups still remain. This calls for a nonrigid (and nonaffine), local transformation model. The FFD model based on B-splines has proven to be a powerful tool when modelling such deformations. In 3D, the FFD is defined by an $n_x \times n_y \times n_z$ mesh of control points $\mathbf{\Phi}$ with spacing $(\delta_x, \delta_y, \delta_z)$. The underlying image is then deformed by manipulating the mesh of control points. The FFD model can be written as the tensor product of the one-dimensional (1D) cubic B-splines

$$\mathbf{T}_{local}(x,y,z) = \sum_{l=0}^{3}\sum_{m=0}^{3}\sum_{n=0}^{3} B_l(u)B_m(v)B_n(w)\phi_{i+l,j+m,k+n} \tag{2}$$

where $i = \lfloor x/n_x \rfloor - 1, j = \lfloor y/n_y \rfloor - 1, k = \lfloor z/n_z \rfloor - 1, u = x/n_x - \lfloor x/n_x \rfloor, v = y/n_y - \lfloor y/n_y \rfloor$ and $w = z/n_z - \lfloor z/n_z \rfloor$

B_r represents the rth basis function of the B-spline

$$\begin{aligned}
B_0(u) &= (1-u)^3/6 \\
B_1(u) &= (3u^3 - 6u^2 + 4)/6 \\
B_2(u) &= (-3u^3 + 3u^3 + 3u + 1)/6 \\
B_3(u) &= u^3/6.
\end{aligned}$$

The similarity metrics tested in this study were Sum of Squared Differences (SSD), Cross Correlation (CC) and Normalised Mutual Information (NMI). In short, NMI outperformed the other two and was therefore used in the remaining experiments. A gradient descent approach was used to optimise the similarity measure. An implementation of the algorithm by Rueckert[6] [4] was applied.

2.3 Atlas construction

An anatomical atlas was constructed from the set of WT mice in an iterative manner using nonrigid registration. The procedure is listed in Table 1.

Lines 6 and 7 from table 1 are intended to reduce the bias towards the choice of reference subject as done with good results in [5]. Figure 2 shows the resulting atlas in three different views and as a surface extracted from the volume.

[6] http://www.doc.ic.ac.uk/~dr/software/

Table 1. Atlas construction

```
1 atlas ← a selected reference subject from the set of WT mice
2 do
3       Nonrigidly register all WT mice to atlas
4       atlas ← Average of all registered mice
5 until atlas stops changing
6 Nonrigidly register atlas to all WT mice
7 Deform atlas by $\bar{\mathbf{d}}$ = the average deformation obtained in step 6
```

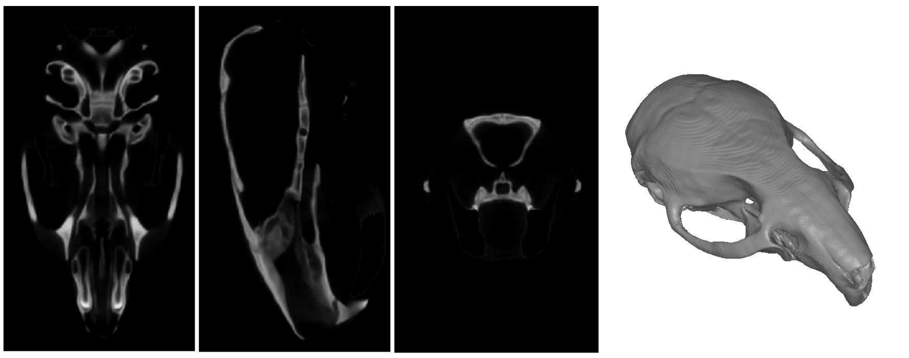


Fig. 2. The craniofacial, nonrigid mouse atlas in (a) axial, (b) sagittal and (c) coronal view. (d): 3D surface view of the atlas.

3 Results

3.1 Registration accuracy

To assess the craniofacial deviations caused by the Crouzon syndrome, the atlas in Figure 2 was registered to all subjects (WT and Crouzon cases). The registration accuracy was examined both qualitatively and quantitatively. Figure 3 shows difference images between one of the Crouzon cases and the atlas before and after registration. Figure 4 shows the closest point difference between the atlas and one of the Crouzon cases before and after registration, as a color overlay on the two surfaces. To provide a quantitative analysis of the registration accuracy, surfaces were extracted from the images and two independent observers put 26 anatomical landmark on all the cases. Using the optimal deformations, the landmarks were also obtained automatically by propagating the atlas landmarks to each of the remaining subjects. Figure 5 shows the landmark errors, i.e. point to point distances between the two observers and between the automatically generated landmarks and each of the observers.

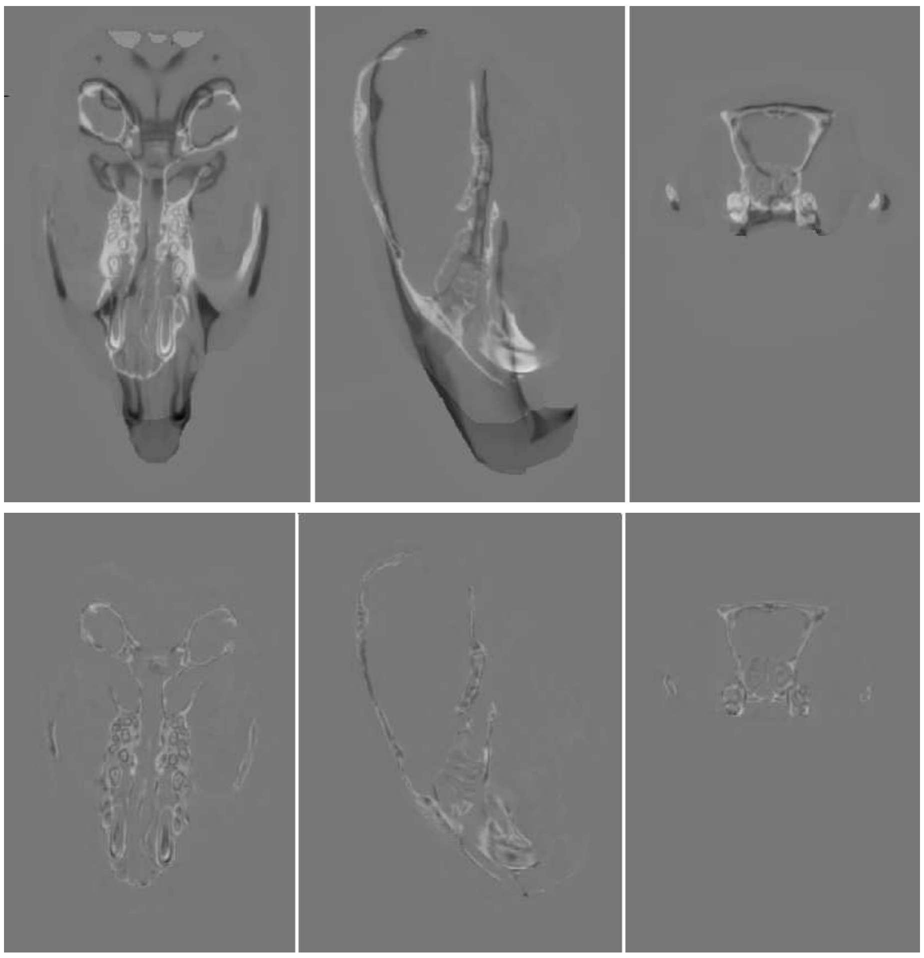
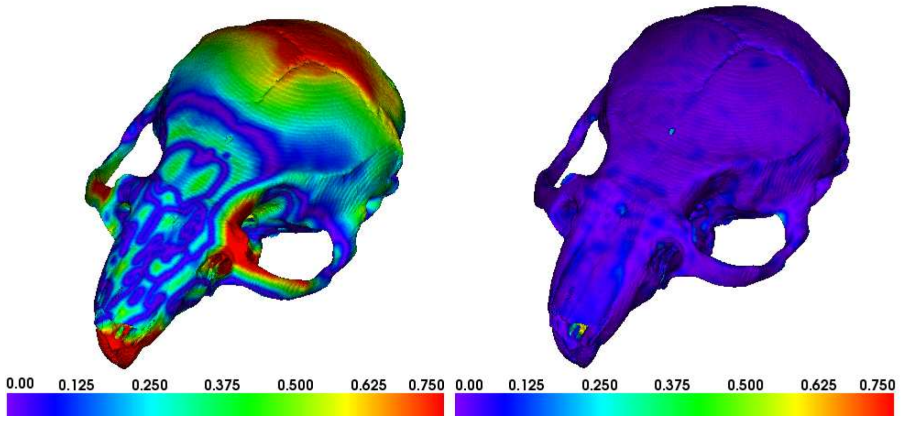

(a) (b) (c)

Fig. 3. Difference images before (top row) and after (bottom row) registration of the atlas to a Crouzon mouse in (a) axial, (b) sagittal and (c) coronal view.

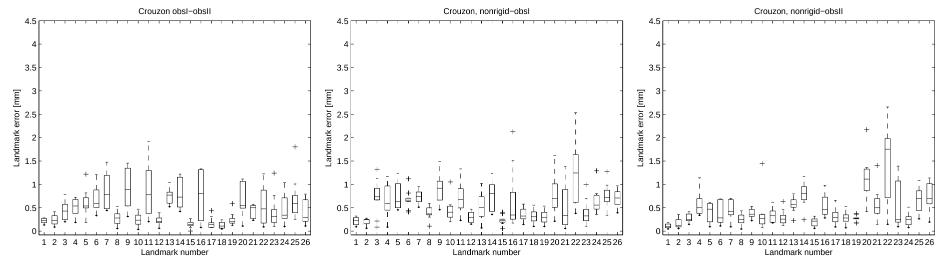
4 Discussion

Figure 3 indicates that the differences between the atlas and the Crouzon case have been compensated for during the registration. In the top row, one can observe large global differences especially in the length and width of the skull. Locally, the shape of the nose and the upper jaw is very different. All these differences and many more, not seen in this figure have been compensated for by the registration (both the global and the local one). Figure 4 gives a semi-quantitative impression of the registration accuracy. Before registration, the distance between the surfaces of the atlas and the Crouzon case reach over 0.75



(a) (b)

Fig. 4. Surface views with color coding denoting closest point difference (in mm) between the surface of the atlas and the Crouzon mouse (a) before and (b) after registration.



(a) (b) (c)

Fig. 5. Landmark errors for Crouzon cases. (a) Interobserver errors, (b)Automatic to observer I, (c) Automatic to observer II.

mm. After affine and nonrigid registration, these differences have been reduced to around 0.1-0.2 mm. A further inspection of the accuracy is given in Figure 5. The landmark errors indicate that the automatic approach is just as good as the manual annotations and even more consistent. However, one landmark (number 22) seems to give problems. This landmark is placed where the frontal nasal suture and sagittal suture meet. The explanation might be that image information is not sufficient for these sutures to match accurately. It was also noted that in some of the Crouzon cases they are hardly visible. Overall, the accuracy is considered to be good. This allows for automatic assessment of the deviations in Crouzon subjects in terms of morphological measuremnts on the skull. Further, the nonrigid registration parameters serve as a good basis for statistical analysis of the deformations between and within the two groups.

5 Conclusion

In summary, this paper has presented the construction of a nonrigid craniofacial wild-type mouse atlas. Furthermore, the atlas has successfully been registered to wild-type mice as well as Crouzon mice. Provided the accurate registrations, it is now possible to automatically assess the growth deviations in Crouzon subjects and carry out statistical analyses of the nonrigid deformations.

References

1. Crouzon, O.: Dysostose craniofaciale héréditère. Bull Mem Sòc Méd Hôp Paris **33** (1915) 545–55
2. Reardon, W., Winter, R.M., Rutland, P., Pulleyn, L.J., Jones, B.M., Malcolm, S.: Mutations in the fibroblast growth factor receptor 2 gene cause Crouzon syndrome. Nat Genet **8** (1994) 98–103
3. Eswarakumar, V.P., Horowitz, M.C., Locklin, R., Morriss-Kay, G.M., Lonai, P.: A gain-of-function mutation of fgfr2c demonstrates the roles of this receptor variant in osteogenesis. Proc Natl Acad Sci, U.S.A. **101** (2004) 12555–60
4. Rueckert, D., Sonoda, L.I., Hayes, C., Hill, D.L.G., Leach, M.O., Hawkes, D.J.: Nonrigid registration using free-form deformations: application to breast MR images. Medical Imaging, IEEE Trans. on **18**(8) (1999) 712–721
5. Rueckert, D., Frangi, A.F., Schnabel, J.A.: Automatic construction of 3D statistical deformation models of the brain using nonrigid registration. IEEE Transactions on Medical Imaging **22**(8) (2003) 1014–1025

Building a real-time digital face, mapping from speech to lip movements and facial expressions

Lars C. Buchhave, Tue Lehn-Schiøler, Mikkel B. Stegmann, and Bjarne K. Ersbøll
Informatics and Mathematical Modelling, Technical University of Denmark
Richard Petersens Plads, Building 321,
DK-2800 Lyngby, Denmark
{lcb, tls, mbs, be}@imm.dtu.dk

Keywords: active appearance model, real-time, speech to video, personal avatar

Abstract

Existing video-footage of a person speaking can be used to construct a model, which maps speech to lip movements and facial expressions. The facial model is based on an Active Appearance Model (AAM) framework (Edwards et al. 1998, Stegmann et al. 2003). The face in the video sequence is matched from frame to frame semi-automatically - typically based on a person-specific model. The matched video sequence contains information on lip movements and other facial expressions, which can be extracted via the parameters in the AAM model parsimoniously. The sound-track containing the speech is analysed in segments corresponding to each video frame. These can now be modelled using e.g. mel cepstral coefficients. Finally, a state-space model is used to map between the speech model and the face model (Lehn-Schiøler 2005).
Once the model and correspondences have been established it is possible to let speech control the lip movements and facial expressions. Because the model is parsimonious and only linear operations involving vectors and matrices are needed it is feasible to make a real-time implementation on a standard computer.

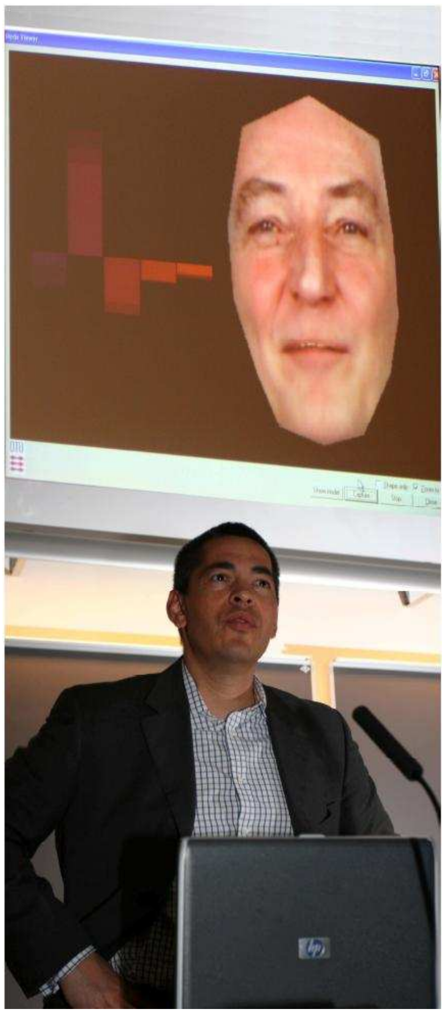


Figure 1. *The digital face preforming at a conference. Picture courtesy of Carsten Broder Hansen, IMM/DTU.*

Introduction

The motivation for transforming a speech signal (or other signals e.g. typed text) into lip movements is obvious for use by handicapped. If video footage of a handicapped person in a period prior to the handicap exists then a personal avatar can be constructed. However, multiple other uses may be considered. Firstly, the language synchronization of movies often leaves the actors mouth moving while there is silence or the other way around, this looks rather unnatural. If it is possible to manipulate the face of the actor to match the actual speech it would be much more pleasant to view synchronized movies (and a lot easier to make cartoons). Secondly, even with increasing bandwidth sending images via the cell phone is quite expensive, therefore, a system that allows single images to be sent and models the face in between would be useful. The technique will also make it possible for hearing impaired people to lip read over the phone. If the person in the other end does not have a camera on her phone, a model image can be used to display the facial movements. Thirdly, when producing agents on a computer (like Windows Office Mr. clips) it would make communication more plausible if the agent could interact with lip movements corresponding to the (automatically generated) speech.

Feature Extraction

Many different approaches have been considered for extraction of sound features. If the sound is generated directly from text (Ezzat and Poggio, 1998), phonemes can be extracted directly and there is no need to process the sound track. However, when a direct mapping is performed one can choose from a variety of features. A non-complete list of possibilities include Perceptual Linear Prediction or J-Rasta-PLP as in (Dupont and Luettin, 2000) and (Brand, 1999), Harmonics of Discrete Fourier Transform as in (McAllister et al., 1998), Linear Prediction Coefficients as in (Lewis, 1991) or Mel Frequency Cepstral Coefficients (MFCC) (Goldenthal et al., 1998) and (Massaro, 1999) and (Hong et al., 2002).
In this work the sound is split into 25 blocks per second (the same as the image frame rate) and a number of MFCC features are extracted from each block. To extract features from the images an Active Appearance model (AAM) (Cootes et al., 1998) is used. The use of this model for lip-reading has previously been studied by Mathews et al. (Matthews et al., 2002). In this work the implementation by Mikkel B. Stegmann (Stegmann, 2002) is used. For the extraction a suitable subset of images in the training set is selected and annotated with points according to the MPEG-4 facial animation standard. Using these annotations a *n*-parameter model of the face is created. Thus, with *n* parameters it is possible to create a photo realistic image of any facial expression seen in the training set. Once the AAM is created the model is used to track the lip movements in the image sequences, at each point the *n* parameters are picked up. In Fig. 2 the result of the tracking is shown for a single representative image.

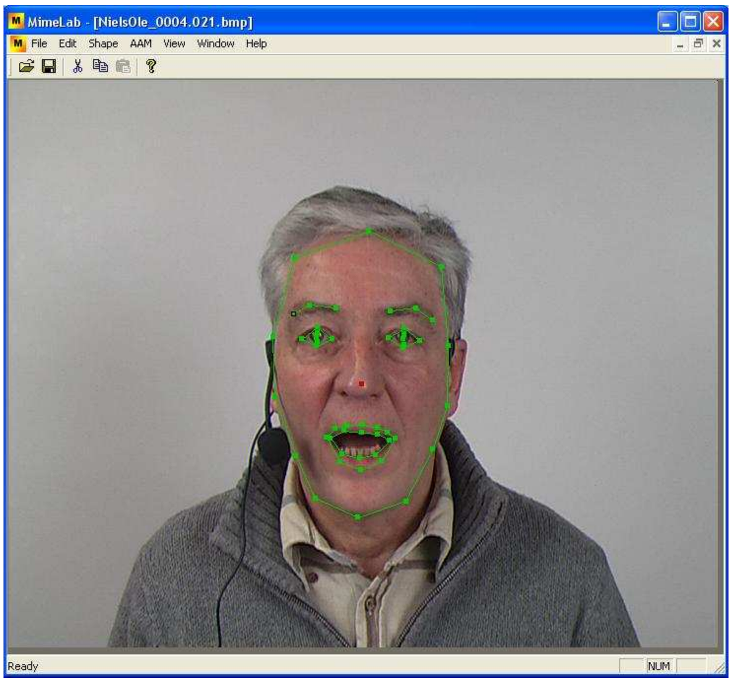

MimeLab - [NielsOle_0004.021.bmp]
File Edit Shape AAM View Window Help
Ready
NUM

Figure 2*. Annotating an image for the AAM model.*

Model

Unlike most other approaches, the mapping in this work is performed by a continuous state space model and not a Hidden Markov Model or a Neural Network. The reasoning behind this choice is that it should be possible to change the parameters controlling the face continuously (unlike in HMM) and yet make certain that all transitions happen smoothly (unlike NN's). Currently an experimental comparison of the performance of HMM's and the continuous state space models is investigated. In this work the system is assumed to be linear and Gaussian and hence the Kalman Filter can be used (Kalman 1960). This assumption is most likely not correct and other models like particle filtering and Markov Chain Monte Carlo are considered. However, as it is shown below, even with the simplification the model produces useful results. The model is set up as follows:

$$x_k = Ax_{k-1} + n_k^x \quad (1)$$
$$y_k = Bx_k + n_k^y \quad (2)$$
$$z_k = Cx_k + n_k^z \quad (3)$$

In this setting z_k is the image features at time *k*, y_k is the sound features and x_k is a hidden variable without physical meaning, but it can be thought of as some kind of brain activity controlling what is said. Each equation has i.i.d. Gaussian noise component n added to it. During training both sound and image features are known, and the two observation equations can be collected in one.

$$\begin{pmatrix} y_k \\ z_k \end{pmatrix} = \begin{pmatrix} B \\ C \end{pmatrix} x_k + \begin{pmatrix} n_k^y \\ n_k^z \end{pmatrix} \quad (4)$$

By using the EM algorithm (Dempster et al., 1977) and (Ghahramani and Hinton, 1996) on the training data, all parameters $\{A, B, C, \Sigma^x, \Sigma^y, \Sigma^z\}$ can be found. Σ's are the diagonal covariance matrices of the noise components. When a new sound sequence arrives Kalman filtering (or smoothing) can be applied to equations (1, 2) to obtain the hidden state *x*. Given *x* the corresponding image features can be obtained by multiplication: $y_k = Cx_k$. If the intermediate smoothing variables are available the variance on y_k can also be calculated.

Real-time Implementation and Model Training

The original implementation of the work was an off-line version where audio clips were used to create a rendered video sequence of the talking model. A real-time version greatly increases the possible uses of the system. Obviously, a real-time implementation needs to be fast, but since all calculations are linear and creation of video by the AAM is also fast, the real-time version proved to be very responsive with no noticeable lagging of the synthesized face.
A sound sampler was integrated and sound packages suitable for the model are created by the sampler continuously. The package is processed by the model and the resulting parameter vector is passed to the AAM which in turn creates the model image for display.
Training the model was done by filming small video clips (~60s) of the test person reading pre-defined sentences. The sentences are gibberish but their objective is to span the phoneme-space of the language being used, thus making the test person pronounce all possible phoneme combinations, e.g. "mm" and "oo" and so on.
The AAM is created by converting each video frame to a still image and ensuring that the model is able to track all facial expressions using an iterative process. If an expression is unable to be tracked by the AAM, the still image is annotated and included in the model. Once the model is sufficiently able to track all expressions, AAM feature vectors are extracted at each frame. MFCCs are also extracted from the audio-tracks in such a

fashion that an MFCC vector is present for each video frame. The audio-to-video model is then trained using these vector sets as previously described.
Training the model proved to be a non-trivial task, however. Firstly, the subject being modelled needs to be very aware of posture and head movement and furthermore needs to deliver the sentences in a normal speaking manner so the program later is able to recognize normal speech by the test subject. Secondly, it is important which video clips are used in the model creation because over and under training of certain words can result in certain facial expressions being over or under emphasized.

Conclusion

A real-time speech to face mapping system relying on continuous state space models is proposed. The system makes it possible to train a unique face model that can be used to transform speech into facial movements. The training set must contain all sounds and corresponding face gestures, but there are no language or phonetic requirements to what the model can handle. The real-time model is very fast and can run smoothly on a normal desktop or laptop computer. However, the resulting simulated face is not always entirely convincing. The major problem is that some phonemes are not emphasised strongly enough although they were present in the training material. Work should be concentrated on improving the actual training of the model to account for these problems. Improvements should also be made on making the audio-to-video model more robust.

Acknowledgement

The COGAIN Network of Excellence is supported by the EU IST 6th framework program. The authors gratefully acknowledge the support from the European Commission in order to be able to present this work.

References

Matthew Brand, “Voice puppetry,” in *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. 1999, pp. 21–28, ACM Press/Addison-Wesley Publishing Co.

T.F. Cootes, G.J. Edwards, and C.J. Taylor, “Active appearance models,” *Proc. European Conference on Computer Vision*, vol. 2, pp. 484–498, 1998.

A. P. Dempster, N. M. Laird, and D. B. Rubin, “Maximum likelihood from incomplete data via the EM algorithm,”*JRSSB*, vol. 39, pp. 1–38, 1977.

S. Dupont and J. Luettin, “Audio-visual speech modelling for continuous speech recognition,” *IEEE Transactions on Multimedia*, 2000.

G.J. Edwards, C.J. Taylor, T.F. Cootes, "Interpreting Face Images using Active Appearance Models", Int. Conf. on Face and Gesture Recognition 1998. pp. 300-305, 1998.

T. Ezzat and T. Poggio, “Mike talk: a talking facial display based on morphing visemes,” *Proc. Compute Animation IEEE Computer Society*, pp. 96–102, 1998.

Z. Ghahramani and G.E. Hinton, “Parameter estimation for linear dynamical systems,” Tech. Rep., 1996, University of Toronto, CRG-TR-96-2.

William Goldenthal, Keith Waters, Thong Van Jean-Manuel, and Oren Glickman, "Driving synthetic mouth gestures: Phonetic recognition for faceme!," in *Proc. Eurospeech '97*, Rhodes, Greece, 1997, pp. 1995–1998.

Pengyu Hong, Zhen Wen, and Thomas S. Huang, "Speech driven face animation," in *MPEG-4 Facial Animation: The Standard, Implementation and Applications*, Igor S. Pandzic and Robert Forchheimer, Eds. Wiley, Europe, July 2002.

Rudolph Emil Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

Dominic W. Massaro, Jonas Beskow, Michael M. Cohen, Christopher L. Fry, and Tony Rodriguez, "Picture my voice: Audio to visual speech synthesis using artificial neural networks," *Proc. AVSP 99*, 1999.

I. Matthews, T.F. Cootes, J.A. Bangham, S. Cox, and R. Harvey, "Extraction of visual features for lipreading," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 2, pp. 198 –213, 2002.

David F. McAllister, Robert D. Rodman, Donald L. Bitzer, and Andrew S. Freeman, "Speaker independence in automated lip-sync for audio-video communication," *Comput. Netw. ISDN Syst.*, vol. 30, no. 20-21, pp. 1975–1980, 1998.

T. Lehn-Schiøler: "Making Faces - State-Space Models Applied to Multi-Modal Signal Processing", Informatics and Mathematical Modelling, Technical University of Denmark, DTU, PhD thesis, 2005.

J. P. Lewis, "Automated lip-sync: Background and techniques," *J. Visualization and Computer Animation*, vol. 2, 1991.

M. B. Stegmann, "Analysis and segmentation of face images using point annotations and linear subspace techniques," Tech. Rep., Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby, Aug. 2002, http://www.imm.dtu.dk/pubdb/p.php?922

M. B. Stegmann, B. K. Ersbøll, R. Larsen, "FAME - A Flexible Appearance Modelling Environment, IEEE Transactions on Medical Imaging", vol. 22(10), Institute of Electrical and Electronics Engineers (IEEE), pp. 1319-1331, 2003.

Classification of fungi using multi-spectral image analysis: A comparison of dimension reductive methods

Line H Clemmensen[1], Michael E. Hansen[2], Bjarne K. Ersbøll[1]

Abstract

Three species of the fungal genus *Penicillium* are subject to classification. In order to obtain an objective identification digital image analysis has been considered. Multi-spectral images of 18 spectra, both visible and NIR, have been acquired. The number of features extracted from the images are many in relation to the number of observations, and the multivariate statistical analysis is hereby complicated. Two methods which solve this problem are compared. One is to use feature selection previous to discriminant analysis, and the other is called LARS-EN (Least Angle Regression - Elastic Net) and was suggested in [1]. LARS-EN can perform both regularization and variable selection or one of these depending on the parameters. LARS-EN is here used for classification and for this use dummy variables representing the three classes are introduced as dependent variables. The methods are compared on their computational effort and on their ability to produce simple models.

References

[1] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *J. R. Statist. Soc. B*, 67(Part 2):301–320, 2005.

[1]IMM, DTU
[2]Center for Microbial Biotechnology, Bio-Centrum, DTU

Estimation of the f/k-factor of Concrete Aggregates by Image Analysis

Peter S. Jørgensen and Bjarne K. Ersbøll

Informatics and Mathematical Modelling, Technical University of Denmark,

Abstract. We analyze images of concrete aggregate materials and extract features by means of a series of erosions and dilations. Feature selection and regression is done simultaneously using the LARS-EN method to estimate the f/k-factor for each aggregate type. Validation is performed using leave-one-out cross-validation .The method proves to be successful in ordering the aggregates according to their f/k-factor for aggregates with grain sizes larger than the pixel size.

1 Introduction

Currently the mix design of self compacting concrete (SCC) is based mainly on trial and error. The f/k-factor is one of many parameters that are part of the SCC mix design model. It is defined as the ratio between the surface area and the volume of a particle with projected diameter 1. The definition leads to for instance spheres having the lowest f/k-factor of 6 and more irregular shapes having larger f/k-factor. The f/k-factor is however hard to estimate on aggregates in practice since the surface area of aggregates can not easily be measured. In the work described here we attempt to estimate the f/k-factor of aggregates using image analysis.

2 Data

The image data consist of various aggregate samples of varying origin, size, color and shape. 8 different aggregates were available. 3 samples from each aggregate type was prepared in separate petri dishes and one multi spectral image of each sample was recorded using the *VideometerLab.* Note that only the exact f/k-factor for the spherical aggregates was known. For the rest of the materials only an ordering of the materials by f/k-factor was available. To have some values to perform regression on, "dummy" values where assigned to the materials indicating the ranking of the relative f/k-factor between them. Thus, the artificial glass spheres were given the lowest ranking of 6 and the very rough surfaced NSV bottom ash was given the ranking 13.

3 Feature Extraction

[1] describes a method for assessing aggregate shapes using silhouette images. From the silhouette images boolean images are constructed as aggregate versus background. A number of binary erosions are performed followed by as many dilations. A measure they call the surface parameter, SP, is calculated as:

$$SP = \frac{A_1 - A_2}{A_1} \tag{1}$$

where A_1 and A_2 are the area of the objects on a boolean image before and after the erosion-dilation cycles. This measure is then used to predict aggregate surface parameters.

By extending the formulation of Equation 1 to grey-scale images a measure similar to the SP can be measured. The change is straight forward and only involves changing the use of binary erosion and dilation above to greys-scale erosion and dilation. This leads to the SP value expressing mainly the degree of changes in surface direction but also to a lesser degree fast changes in the direction of grain edges in the image plane.

The proposed feature was calculated for the combinatorial space spanned by varying the parameters over a range of values. The number of erosions/dilations was set at $1-8$ and the disc shaped structuring element was set at radii of $1-25$. Combined this resulted in 392 features.

4 Feature Selection and Regression

The feature space obtained from measuring the SP is very high dimensional with a high degree of collinearity between features. Obviously some form of feature selection will be needed to not allow all features into the regression model. The LARS-EN [2] (Least Angle Regression - Elastic Net) method is used to perform simultaneous feature selection and regression. It is expected that many features will have almost exactly the same values and thus the exact features that are included in the model will vary depending on which samples are included when training the model.

5 Results

Figure 1 shows the result of leave-one-out cross validation using samples with medium and large grain sizes. There is generally a good separation of aggregates with differing f/k-factor orderings, and the estimates do no deviate greatly from their target values.

When comparing aggregates of the same type with differing grain size distributions, for instance the 14mm and the 6mm glass spheres (*glas14* and *glas6*), it is seen that the same predicted f/k-value is achieved for both size distributions. This indicates that the model is largely independent of the size distribution of aggregates.

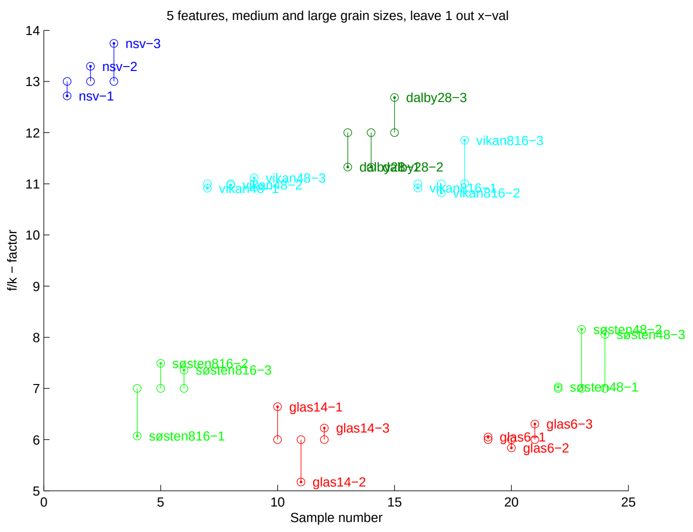


Fig. 1. The result of leave-one-out cross validation using 5 features and large and medium grain sizes. The line at each name shows the distance from the predicted value (where the name is located) to the target value.

Different features were included in the models by the LARS-EN algorithm depending on which samples were omitted from the training set during the cross-validation. This indicates that it is a group of features that are of interest rather than a few specific features. Recall that the features used were variations of different degrees of smoothing, erosions/dialtions and structuring element size. In general the features without smoothing and with a modest amount of morphological change performed the best.

6 Conclusion

There are good indications that a good prediction of f/k-factor can be obtained from the proposed features on aggregates with grains that are clearly distinguishable on the images. The model is able to order the aggregates according to their true f/k-factor value and is largely independent of grain size. Overall the amount of completely different samples (not just repetitions) is on the low end to draw any decisive conclusions but the trend that has been shown here is promising.

Acknowledgements

We would like to thank the Danish Technological Institute for providing the aggregate materials and general knowledge of SCC.

References

1. Eyad Masad and Joe W. Button. Unified imaging approach for measuring aggregate angularity and texture. *Computer-Aided Civil and Infrastructure Engineering*, 15:273–280, 2000.
2. Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Royal Statistical Society*, 67:301–320, 2005.

Classification of biological objects using Active Appearance Modelling *Extended Abstract*

Anders Bjorholm Dahl   Henrik Aanæs   Rasmus Larsen   Bjarne Ersbøll
Informatics and Mathematical Modelling, DTU
abd@imm.dtu.dk

Abstract

Here the use of the popular active appearance models (AAM) for classification of biological objects is investigated, based on images. Two approaches are investigated; one where an AAM is fitted to all classes in question, and one where an AAM is fitted to each individual class. In the first case, the parameters for the 'global' model is used as a feature vector for a given instance, which again is feed into a linear classifier. In the latter all the AAM's are fitted to a new object in question, and the goodness of fit of the models are used as a classifier. The two approaches are compared on two real data sets, one containing various vegetables and one containing different species of wood logs.

1 Introduction

Object recognition is one of the fundamental problems in computer visions, and plays a vital role in constructing 'intelligent' machines, in that is needed for a robot to do semantic reasoning about the world it 'sees' in general. Object recognition, however, also plays a more mundane role, in that it is a vital part of many visual inspection systems, e.g. determining what produce is passing through a production line or which person trying to board an aeroplane. Our initial motivation for this work is the construction of an automated forestry system, which needs to keep track of wood logs.

Many of the objects in our daily environment in general, and in our motivating problem in particular, are biological, and pose special challenges to a computer vision system. The origin of these challenges are the high degree of interclass variation, which we as humans are very good at dealing with. Consider e.g the multitudes of ways a face or a potato can look. In this regard AAM's have proven very well suited for addressing this challenge in the case of image registration, c.f. e.g. [1]. It is thus highly interesting if this property of the AAM's also transfer to object recognition, and how this should be achieved/implemented. This is the scope of this paper.

Previously the use of AAM's for object recognition have been considered. Notably, Edwards et al. [2] use AAM's for face recognition. Here the use of AAM's for object recognition on general biological objects is investigated. The necessitates considering implementation issues, such as if one global AAM should be used or one for each class. An issue which is also addressed here.

2 Classification Methods

The AAM model - in 2D as it will be used here - is a description of an object in an image via it's contour or shape and it's texture. Each of these entities can be represented as a vector, i.e. $\mathbf{s}_i$ and $\mathbf{t}_i$ respectively, where the subscript, i, denotes an instance. The parameters of the AAM model is, however, a lower dimensional vector, $\mathbf{p}_i$, and a specific AAM consists of an affine mapping for $\mathbf{p}_i$ to $\mathbf{s}_i$ and $\mathbf{t}_i$, i.e.

$$\mathbf{m}_i = \begin{bmatrix}\mathbf{s}\\ \mathbf{t}\end{bmatrix}_i = \mathbf{\Phi}\mathbf{p}_i + \mu \ , \tag{1}$$

where $\mathbf{\Phi}$ and μ are a matrix and vector respectively representing the affine map. Typically, the AAM or $\mathbf{\Phi}$ is estimated from an annotated training set. The variance structure of the $\mathbf{m}_i$ is estimated, followed by a PCA on this structure to have a sufficiently sparse model. The interested reader is referred to [1], for a more detailed description.

The classification problem under consideration is, that we are given images with objects x_i, belonging to one of n classes, i.e. $x_i \in \mathcal{C}_j \;\; j \in [1 \dots n]$, and the class is to be determined, i.e. j.

In the following the two methods of interest will be described.

2.1 Global AAM as Classifier

In this case a single global AAM, $\mathbf{\Phi}, \mu$, is fitted to instances of all classes. Following this, the $\mathbf{p}_i$ are calculated for each instance in the training set, which, together with the corresponding class, is used in constructing a standard linear classifier, c.f. e.g. [3].

Given the image of an unknown object, the $\mathbf{m}_i$ and then the $\mathbf{p}_i$ are calculated which are then feed into the above mentioned classifier, for a result.

Interesting issues, in this regard, apart from the straight forward performance, is the construction of the training set. E.g. what is the effect of one class being highly over-represented, or how well would an AAM trained on potatoes be at at distinguishing between carrots and cucumbers.

2.2 Multiple AAM's as Classifiers

In this case an AAM, $\mathbf{\Phi}_j, \mu_j$, is fitted to each class $\mathcal{C}_j$. I.e. the training set is divided into its component classes, and one AAM is fitted to each.

In this case the classifier proposes a goodness of fit, i.e. for a given unknown object x_i a score, s_{ij}, is calculated for each class, using a Mahalanobis classifier as follows

$$s_{ij} = \left(\mathbf{\Phi}_j\mathbf{p}_i - \mu_j\right)^T \Sigma_j^{-1} \left(\mathbf{\Phi}_j\mathbf{p}_i - \mu_j\right) \log(\det(\Sigma_j)) \ , \tag{2}$$

where Σ_j is the variance structure associated with $\mathcal{C}_j$. Object x_i is then classified as the class with the lowest score, s_{ij}. This corresponds to classifying x_i as the class where it is most probable.

3 Experiments

The above posed questions are investigated via experiments on classifying carrots vs. potatoes and classifying three common tree species, see figure 1 and 2. The experiments are conducted using the AAM-API by Stegmann [4] [5].

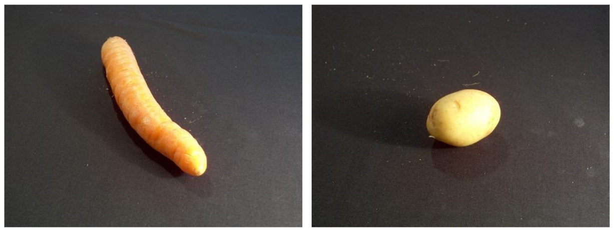


Figure 1: Example of images belonging to the two classes in the carrots vs. potatoes experiment.

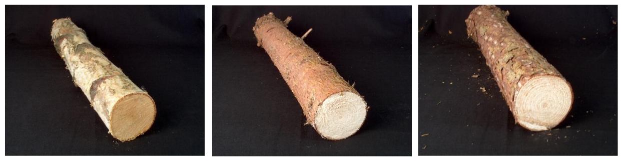


Figure 2: Example of images belonging to the three classes in the wood log experiment.

We have made a combined AAM for each of the two combined experiments and an AAM for each class in the two multiple AAM experiments. These model are build from annotated images, see figure 3. For each model a representative sample of images are used for building the model. To be representative the images are selected, so they represent most of the variation for the class. E.g. carrots vary in shape and texture, so the models containing carrots are build from images showing both the different shapes, e.g. long and thin vs. short and thick carrots, and difference in texture, e.g. very yellowish vs. more greyish carrots.

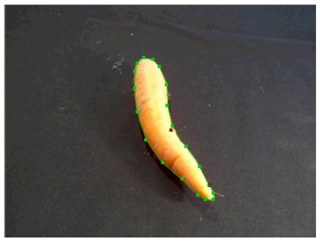


Figure 3: Example of an annotated image used for building the AAM.

We have tested the model on series of unknown images. At first the AAM is fitted to the object in the unknown image by calculating the model parameters $\mathbf{p}_i$ that gives the least difference in the model instance and the image, see figure 4. The parameters obtained by this minimization are secondly used for classification, both in the global and the multiple case.

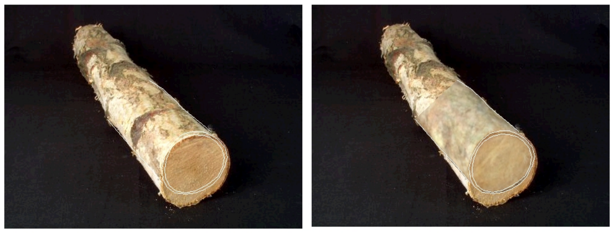


Figure 4: Example of a model fitted to an unknown image (left) and the model instance overlaid (right).

References

[1] T. F. Cootes and C. J. Taylor. Statistical models of appearance for medical image analysis and computer vision, 2004. In Proc. SPIE Medical Imaging.

[2] G. J. Edwards, T. F. Cootes, and C. J. Taylor. Face recognition using active appearance models. *Computer Vision - ECCV'98. 5th European Conference on Computer Vision. Proceedings*, pages 581–95 vol.2, 1998.

[3] T. Hastie, J. Tibshirani, and J. Friedman. *The Elements of Statistical Learning, Data Mining, Inference and Prediction.* Springer, 2001.

[4] M. B. Stegmann. The AAM-API, 2003. Platform: MS Windows.

[5] M. B. Stegmann, B. K. Ersbøll, and R. Larsen. FAME - a flexible appearance modelling environment. *IEEE Transactions on Medical Imaging*, 22(10):1319–1331, 2003.