THE QUEEN'S AWARDS
FOR ENTERPRISE
2001

BBC

R&D White Paper

WHP 083

March 2004

Bi-directional affine motion compensation using a content-based, non-connected, triangular mesh

Marc Servais[*], Theo Vlachos[*] and Thomas Davies[†]

[*]University of Surrey, UK; and [†]BBC Research and Development, UK

Research & Development
BRITISH BROADCASTING CORPORATION

BBC Research & Development
White Paper WHP 083

Bi-directional affine motion compensation using a content-based, non-connected, triangular mesh

Abstract

A new mesh-generation scheme for motion compensation is presented and discussed. To start with, spatio-temporal segmentation is applied to an image (and a reference frame) in order to identify the boundaries of moving regions, which are then approximated with polygons. A triangular mesh is generated within each polygon, thus ensuring that no triangle straddles multiple regions. The affine motion parameters are then estimated separately for each triangle, using bi-directional motion estimation. Results for various test sequences demonstrate the advantages offered by the use of a contentbased mesh, and by affine motion compensation.

This document was originally presented at the 1st European Conference on Visual Media Production (CVMP), 15-16 March 2004 at The IEE, Savoy Place, London.

© BBC 2004. All rights reserved.

White Papers are distributed freely on request.

Authorisation of the Chief Scientist is required for publication.

© BBC 2004. All rights reserved. Except as provided below, no part of this document may be reproduced in any material form (including photocopying or storing it in any medium by electronic means) without the prior written permission of BBC Research & Development except in accordance with the provisions of the (UK) Copyright, Designs and Patents Act 1988.

The BBC grants permission to individuals and organisations to make copies of the entire document (including this copyright notice) for their own internal use. No copies of this document may be published, distributed or made available to third parties whether by paper, electronic or other means without the BBC's prior written permission. Where necessary, third parties should be directed to the relevant page on BBC's website at http://www.bbc.co.uk/rd/pubs/whp for a copy of this document.

BI-DIRECTIONAL AFFINE MOTION COMPENSATION USING A CONTENT-BASED, NON-CONNECTED, TRIANGULAR MESH

Marc Servais*, Theo Vlachos* and Thomas Davies†

*University of Surrey, UK; and †BBC Research and Development, UK

ABSTRACT: A new mesh-generation scheme for motion compensation is presented and discussed. To start with, spatio-temporal segmentation is applied to an image (and a reference frame) in order to identify the boundaries of moving regions, which are then approximated with polygons. A triangular mesh is generated within each polygon, thus ensuring that no triangle straddles multiple regions. The affine motion parameters are then estimated separately for each triangle, using bi-directional motion estimation. Results for various test sequences demonstrate the advantages offered by the use of a content-based mesh, and by affine motion compensation.

1 INTRODUCTION

Motion Estimation and Compensation have traditionally been performed using block-based methods. They offer the advantage of being fast, easy to implement and fairly effective over a wide range of video content.

However, block-based approaches suffer from two drawbacks. First, they are typically used to estimate only translational motion, and thus cannot accurately model more complex types of motion (such as rotation and zoom).[1] Secondly, the size and shape of blocks is fixed, and consequently independent of scene content. Thus a block covering two regions with different motion will not be able to accurately model the motion within both regions simultaneously.

Triangular and quadrilateral 2D meshes were first proposed as an alternative to block-based systems in order to allow for affine motion estimation [2, 3] and thus achieve better adaptation to moving regions . As each node within a mesh moves, so it causes the triangles of which it is a vertex to warp. This warping effect allows for more complex motion to be modelled.

In most early mesh implementations, nodes were placed at regular intervals throughout the image. However it was shown to be advantageous (and intuitively plausible) to position the nodes along object boundaries, or even to have a separate mesh for each foreground object [4]. The use of meshes for object-based video was encouraged by the development of the MPEG-4 standard, in which Video Object Planes are used to represent arbitrarily shaped regions within a scene [5]. One disadvantage of using an object-based mesh is that the boundary of each object needs to be specified, resulting in a significant overhead. In practice, regions are often approximated using polygons/splines or from region boundaries in preceding frames.

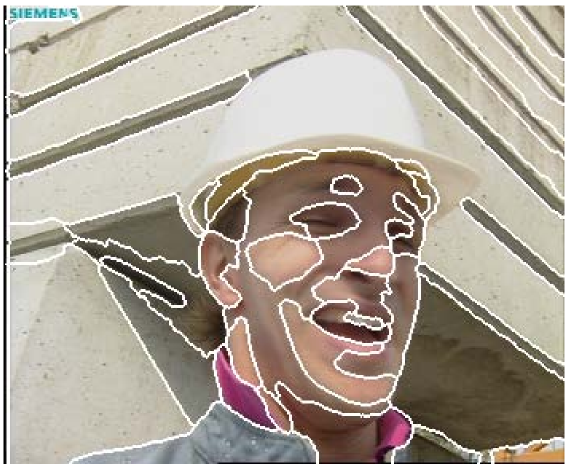
A significant problem with using fully-connected meshes for motion compensation is that neither occlusion nor uncovering can be accurately modelled. One proposed solution involves causing the mesh to *rip* or tear along occlusion boundaries, and allowing overlapping of the mesh along these tears [6, 7]. Another solution is based on first identifying the *background to be covered* within a frame, and allowing no nodes to be placed there. In addition, a *model failure* region is detected and the mesh is refined inside this region [8]. Both of the above approaches were demonstrated to perform well for “head and shoulder” type sequences.

The approaches outlined above employ motion compensation of the current frame from *one* previously encoded reference frame. This is similar to an MPEG P-frame (with the obvious exception that mesh-based motion compensation is used). Recently, a bi-directional mesh for video objects has also been developed, which allows for motion compensation from two previously encoded reference frames, as in the case of MPEG B-frames [9].

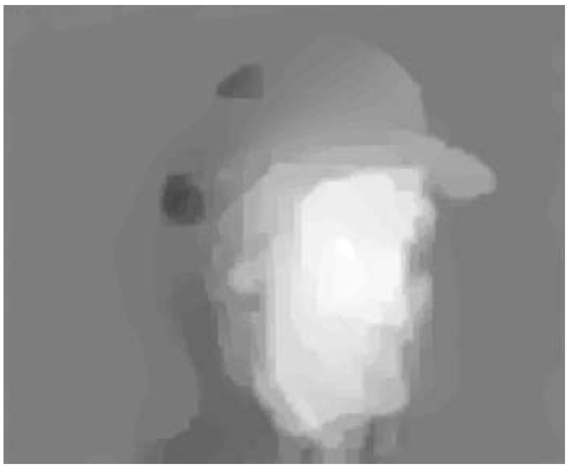
This paper describes an implementation of bi-directional affine motion estimation using a content-based, non-connected mesh. When designing a content-based mesh, the first step usually involves segmenting a scene or image into regions or into video objects. The MPEG-4 standard allows for region-based coding, but does not describe which segmentation method should be used, since this is left up to the encoder. A wide variety of spatio-temporal segmentation methods have been proposed [10, 11, 12] and this is currently a very active area of research. However, an intuitively simple method based on existing colour segmentation and dense motion estimation tools was developed in order to segment each frame. (The approach is outlined in Section 2.1.)

[1] One exception is the use of *generalised block matching* [1], which allows blocks to be deformed using affine, perspective or bilinear transforms.

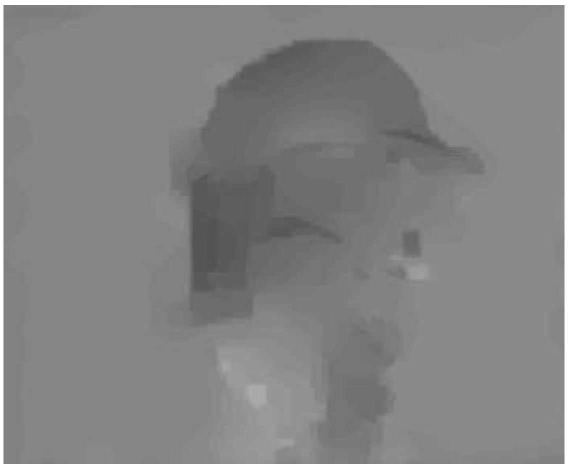
(a) Frame 13 of the “Foreman” sequence



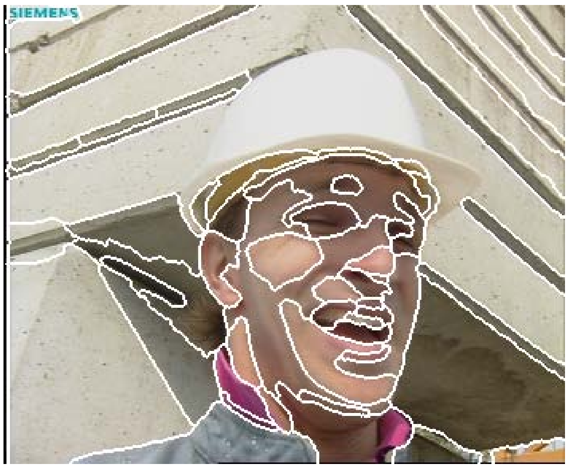
(b) After JSEG spatial segmentation



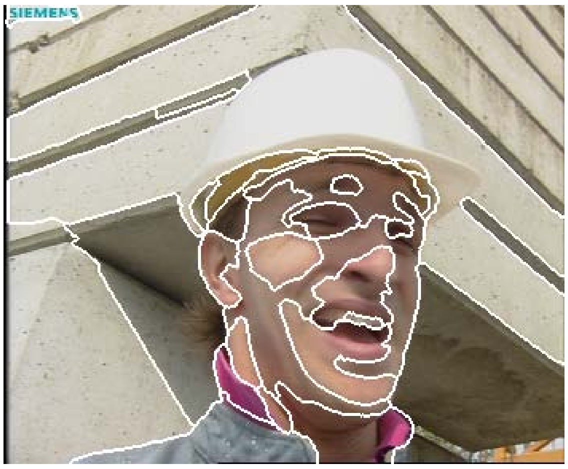
(c) Representation of the horizontal dense motion-vector field (frame 13 relative to frame 11); black: left, white: right



(d) Representation of the vertical dense motion-vector field (frame 13 relative to frame 11); black: down, white: up



(e) After region splitting



(f) After region merging

Figure 1: An example of Spatio-temporal segmentation using region splitting and merging, applied to frame 13 of the “Foreman” sequence. Note that motion has been calculated relative to frame 11.

Each frame is assumed to have two reference frames: one preceding it and one subsequent to it in time. Once a segmentation map has been obtained for the current frame, regions in the frame are approximated with polygons. A novel polygon approximation algorithm is presented, designed to ensure that neighbouring polygons share a common boundary (Section 2.2). A triangular mesh is then created within each polygon-shaped region (Section 2.3), resulting in the current frame being fully covered by triangles.

Following this, the translational and affine motion of each triangle in the current frame are estimated relative to the two reference frames (Section 3). Note that in the current implementation a *non-connected* mesh is used. This means that the affine motion parameters are estimated separately for each triangle, and that the motion of one triangle does not influence that of its neighbours.

Finally, experimental results after the motion compensation stage are presented for a variety of test sequences (Section 4). The performance when using a content-based mesh is compared to that of a regular triangular mesh, and the observations are discussed.

2 MESH GENERATION

2.1 Spatio-Temporal Segmentation

Spatio-temporal segmentation methods divide a scene into regions that differ in both motion and spatial characteristics (e.g. colour, texture and intensity). However, from the point of view of motion compensation for video coding applications, it is not necessary to segment a group of objects that are moving similarly (and thus have the same motion properties). Consequently, regions only need to be segmented if they have different motion properties.

Since the primary focus of this research is not segmentation, it was decided to use two established schemes to obtain reasonable segmentation results across a range of different sequences: JSEG [13] is used to achieve spatial segmentation, and Black and Anandan's software [14] is used to estimate dense optical flow.

The segmentation process then operates as follows (with Figure 1 illustrating the segmentation process for frame 13 of the "Foreman" sequence):

a) Perform spatial segmentation of the current frame using JSEG. (Figure 1(b) shows the regions obtained after spatially segmenting frame 13 of "Foreman".)

b) Estimate the dense motion-vector field, relative to the closer of the two reference frames. (Figures 1(c) and 1(d) provide a representation of the dense motion vector field - both the horizontal and vertical components.)

c) Split stage: If a region has a high motion vector variance then segment it spatially using JSEG. (Figure 1(e) shows the result after these regions have been further segmented.)

d) Merge stage: If two neighbouring regions have similar mean motion vectors *and* their joint variance is low, then merge the two regions. (Figure 1(f) illustrates the result of merging regions with similar motion.)

2.2 Polygon Approximation

Once the current image has been segmented into regions, it is necessary to approximate each region with a polygon. However, a simple polygon approximation of each region in turn is likely to result in the new (polygon-shaped) regions not being correctly aligned. This is because polygons corresponding to neighbouring regions will in general either overlap or leave "holes" along their common boundary.

As a result, a different polygon approximation strategy (related to that in [8], but based on neighbouring regions) is proposed:

a) For each pair of neighbouring regions, find the common boundary portion(s). For example, Figure 2 shows two neighbouring regions which share a common boundary portion. This is the curved line from P_a to P_b along their common border.

b) Initially, the polygon approximation points for this curved segment consist of the two endpoints P_a and P_b.

c) For each common boundary portion, move from the

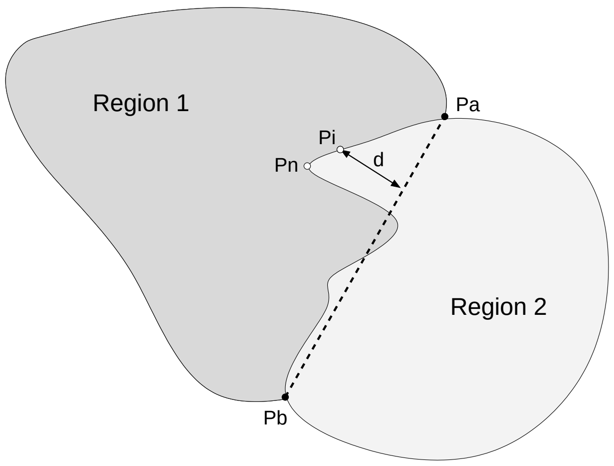


Figure 2: Using line segments to approximate the boundary between two regions

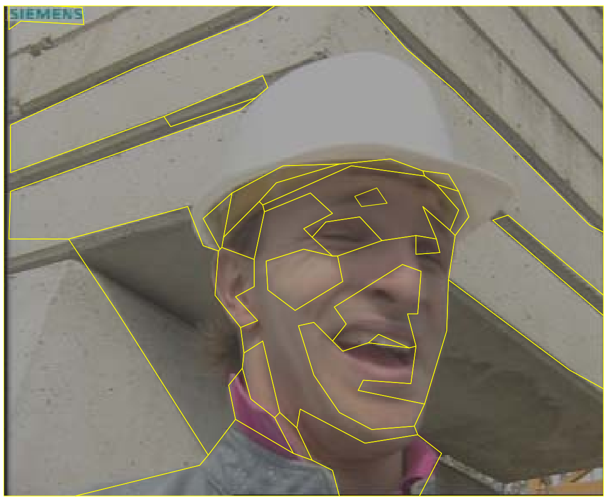


Figure 3: Polygon approximation of regions in Fig 1(f)

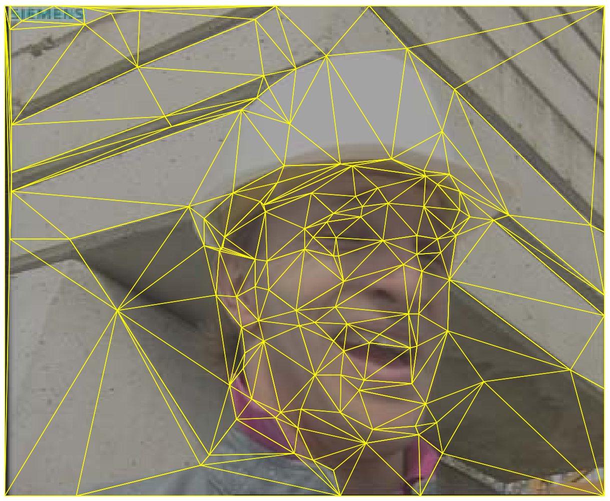


Figure 4: Triangulation of the polygons in Figure 3

beginning (P_a) to the end (P_b). Let P_i be the current point along the boundary. If the perpendicular distance, d, between P_i and the straight line connecting P_a and P_b exceeds some threshold (d_{max}), then a new polygon approximation point needs to be chosen.[2] Similarly, If the *area* between the boundary curve and the straight line from P_a to P_b exceeds some threshold (A_{max}), then a new polygon approximation point needs to be chosen.[2]

d) If a new polygon approximation point needs to be chosen, it is selected as that point along the curve (from P_a to P_b) which has the maximum perpendicular distance, d, from the straight line from P_a to P_b. (Call this new point P_n.) The above process is then applied recursively to the boundary curves from P_a to P_n and from P_n to P_b.

e) Otherwise, if no new polygon approximation point is required, the straight line from P_a to P_b is considered an adequate approximation of the curved boundary between these two points.

After applying this process to each pair of neighbouring regions, a polygon approximation of all regions in the current image is obtained, as illustrated in Figure 3.

2.3 Triangulation

Following the approximation of each region with a polygon, the next step is to create a triangular mesh within each polygon. This is done by using the scheme described in [15] to generate a Delaunay mesh inside each polygon, and forcing the edges of the polygon to be part of the mesh.[3] Figure 4 shows the results once triangulation has been applied to all (polygon-shaped) regions.

3 MOTION ESTIMATION & COMPENSATION

3.1 Translational Motion

The motion parameters for each triangle within the current frame are estimated independently of those for other triangles. This is possible because the mesh is non-connected. The translation vector is determined as follows:

a) Consider all the points within a given triangle in the current frame (such as the one depicted in Figure 5(a).

b) Match these points to the corresponding set of translated points within the reference frame. Note that the translation is restricted to some specified search radius, as illustrated in Figure 5(b).

c) Matching is performed by calculating the *mean square error* (MSE) between the (colour or greyscale) intensities of the points in the current frame and the intensities of the translated points in the reference frame. (Alternatively, the mean absolute difference or some other appropriate metric may be used.)

d) The shift which results in the smallest MSE is chosen as the translation vector, (u_t, v_t), for the triangle under consideration.

[2]Values of $d_{max} = 5$ and $A_{max} = 128$ were found empirically to produce reasonable results for a variety of CIF and SIF resolution sequences. The values can be reduced if a more precise polygon approximation is required, or increased if fewer nodes are desired.

[3]The polygon boundaries are specified as segments within a *planar straight line graph*, and (where possible) triangles are created with all angles larger than 20 degrees. Up to four interior nodes per polygon are allowed to be added during the triangulation process.

e) Note that translation vectors are calculated to integer pixel accuracy. Sub-pixel accuracy can be achieved during the affine motion estimation stage.

f) Two reference frames are used - one before and one after the current frame. The one which gives rise to a smaller translation motion compensation error is selected as the sole reference frame for the affine motion estimation stage.

3.2 Affine Motion

Following the estimation of translational motion, the (translated) triangle in the reference frame is warped slightly to see if an even better match to the original triangle in the current frame can be obtained. This warping is performed using the six-parameter affine model, as illustrated in Figure 5(c) and outlined below:

a) Each of the three vertices of the (translated) triangle in the reference frame is moved within a small search area, while keeping the other two vertices constant.

b) This results in a warped triangle, which is matched to the original in the current frame. Once again, matching is achieved by minimising the motion compensated MSE for points inside the triangle.

c) For each of the three triangle vertices in the *current* frame, $\{(x'_i, y'_i) : i \in \{1,2,3\}\}$, the corresponding vertices in the *reference* frame, $\{(x_i, y_i)\}$ are related according to the equation:

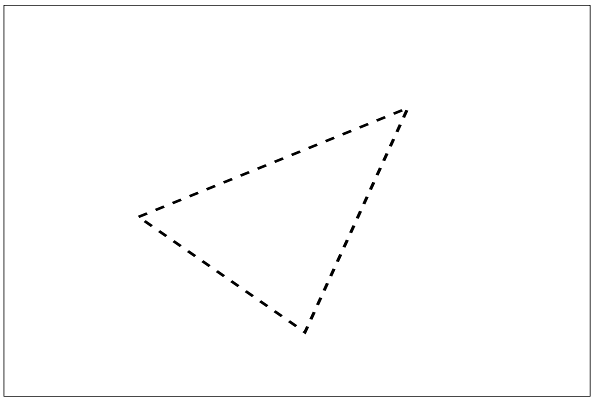
$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} x'_i \\ y'_i \end{bmatrix} + \begin{bmatrix} u_t \\ v_t \end{bmatrix} + \begin{bmatrix} u_i \\ v_i \end{bmatrix} \qquad (1)$$

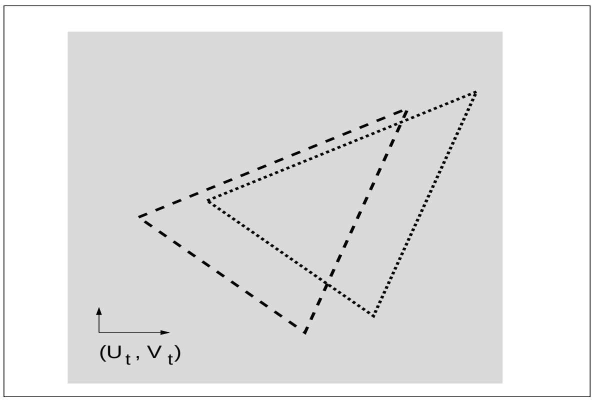
where (u_t, v_t) is the (optimal) translation motion vector calculated during the previous stage, and $\{(u_i, v_i) : i \in \{1,2,3\}\}$ are the *additional* displacements of the three triangle vertices in the reference frame.

d) During the affine motion estimation stage, (u_i, v_i) is varied within a small search range for each of the three nodes. Note that consequently $\{(x_i, y_i)\}$ is known for each such variation. This is because $\{(x'_i, y'_i)\}$ and (u_t, v_t) are constant for each triangle, the latter having been determined during the translation estimation stage.

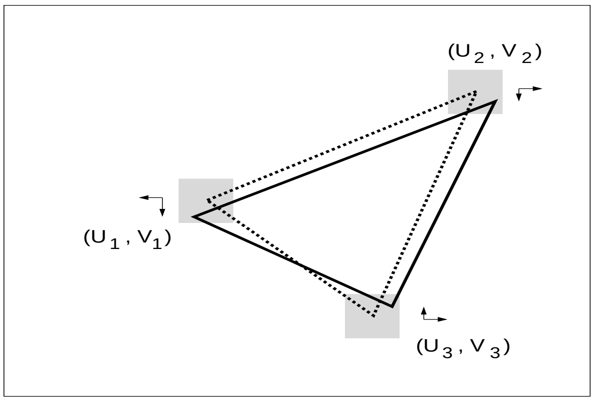
e) In general, for a pixel (x', y') in a triangle in the current frame, the corresponding point (x, y) in the reference frame is given by the affine transform:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 + u_t \\ a_4 & a_5 & a_6 + v_t \end{bmatrix} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \qquad (2)$$


(a) The position of a triangle in the *current* frame (shown with dashed lines).



(b) The dotted lines show the position of a triangle in the *reference* frame that is a translated version of the triangle in (a). Note that the initial matching is performed using translation only. The translation search region is shown in grey and the resultant motion vector, (u_t, v_t), is also depicted.



(c) The solid lines show the position of the triangle in the *reference* frame that is a translated and warped version of the triangle in (a). In this case, the initial matching has been performed using translation, and further refined with affine warping. The areas shaded grey show the search range for each vertex when estimating the affine motion. The three corresponding motion vectors for each vertex are also represented.

Figure 5: Translation and Affine Motion Estimation

where $\{a_1, \ldots, a_6\}$ are the six affine motion parameters. These can be determined by considering Equation 2 in the case of the three triangle vertices. This yields the system of equations:

$$\underbrace{\begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ x_3 \\ y_3 \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} x'_1 & y'_1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x'_1 & y'_1 & 1 \\ x'_2 & y'_2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x'_2 & y'_2 & 1 \\ x'_3 & y'_3 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x'_3 & y'_3 & 1 \end{bmatrix}}_{\mathbf{B}} \underbrace{\begin{bmatrix} a_1 \\ a_2 \\ a_3 + u_t \\ a_4 \\ a_5 \\ a_6 + v_t \end{bmatrix}}_{\mathbf{a}} \tag{3}$$

for which the positions of the vertices $\{(x'_i, y'_i)\}$ and $\{(x_i, y_i)\}$ are known. Solving for $\mathbf{a}$ yields:

$$\mathbf{a} = \mathbf{B}^{-1}\mathbf{x} \tag{4}$$

Motion compensation of any point in the current triangle can then be performed by substituting the affine motion parameters from $\mathbf{a}$ into Equation 2.

The above affine motion estimation process does not search the entire subspace of affine parameters. This is because each triangle vertex is moved individually within its search region (while the other two vertices are fixed). A full search involves moving the three vertices simultaneously across the range of possible positions (within the search radius). However, such a full search is computationally expensive and is $O(r^6)$, where r is the search range. In contrast, the sub-optimal search (i.e. moving one vertex at a time) is significantly faster, being $O(r^2)$. A somewhat more robust sub-optimal affine motion estimation approach involves moving vertices one, two and three of the triangle, followed by a second perturbation of vertex one. (This was the method used in the research described in this paper.)

When applying motion compensation to a pixel position (which has integer co-ordinates), the resulting co-ordinates in the reference frame are (in general) non-integer real numbers. In this case, bi-linear interpolation (from the four neighbouring pixels) is used to estimate the intensity at the desired point in the reference frame. The use of bi-linear interpolation also allows affine motion vectors to be calculated with sub-pixel accuracy.

The affine motion of a particular triangle is described in terms of six parameters. These can either be the $\{a_1, \ldots, a_6\}$ from Equation 2, or the motion vectors of the three vertices, namely the $\{(u_i, v_i) : i \in \{1, 2, 3\}\}$ from Equation 1. In a practical coding system it is often easier to use the latter, since the three motion vectors can be specified (or quantised) with equal precision.

Finally, it should be noted that the matrix $\mathbf{B}^{-1}$ in Equation 4 needs to be calculated only once for each triangle during the motion estimation stage. This is because the only variables $\mathbf{B}$ contains are the co-ordinates of the triangle in the *current* frame, which do not vary.

4 EXPERIMENTAL RESULTS

The performance of mesh-based motion compensation was evaluated for three different sequences, using both regular and content-based meshes. The test sequences used ("Foreman", "Flower Garden" and "Football") all contain significant foreground and background motion. When estimating translational motion, a search radius of ± 63 pixels was used, with an additional search radius of ± 3 pixels for affine motion.[4]

Figure 6 shows the results of using both a content-based and a regular mesh for a frame of the "Foreman" sequence. As expected, the PSNR increases as the number of triangles in the mesh is increased. The advantage offered by affine motion compensation (over the purely translational case) is clearly evident, corresponding to a gain of approximately 2 dB. It can also be seen that the use of a content-based mesh provides superior performance to that of a regular mesh with the same number of triangles. As shown in the graph, these two observations hold across the range of different mesh sizes that were tested.

Figures 7 and 8 illustrate results for 25 frames from the "Flower Garden" and "Football" sequences. Each compensated frame ($\tilde{I}_n$) has been motion compensated from two original frames (I_{n-2} and I_{n+2}) preceding and subsequent to it in time. (The distance between the current frame and the two reference frames was chosen as two frame intervals in either direction).

For both of the sequences, the use of a content-based mesh (with affine motion) offers an average improvement of greater than 1 dB over a standard block-based (translation only) approach, and a mean improvement of more than 0.4 dB over a regular triangular mesh (with affine motion).[5]

These results suggest that the shape and layout of triangles in a content-based mesh offer a significant advantage over uniformly spaced triangles or blocks. This is because triangle boundaries in the content-based mesh are designed so as not to straddle the boundaries of moving

[4]In the case of "Foreman", motion vectors were estimated to pixel accuracy, while half-pixel accuracy was used for the other two sequences.

[5]Note that for each frame, the content-based mesh was created first, and following this a regular mesh with a similar number of triangles was generated. The number of triangles was also used to determine the (approximate) number of blocks for the block-based approach.

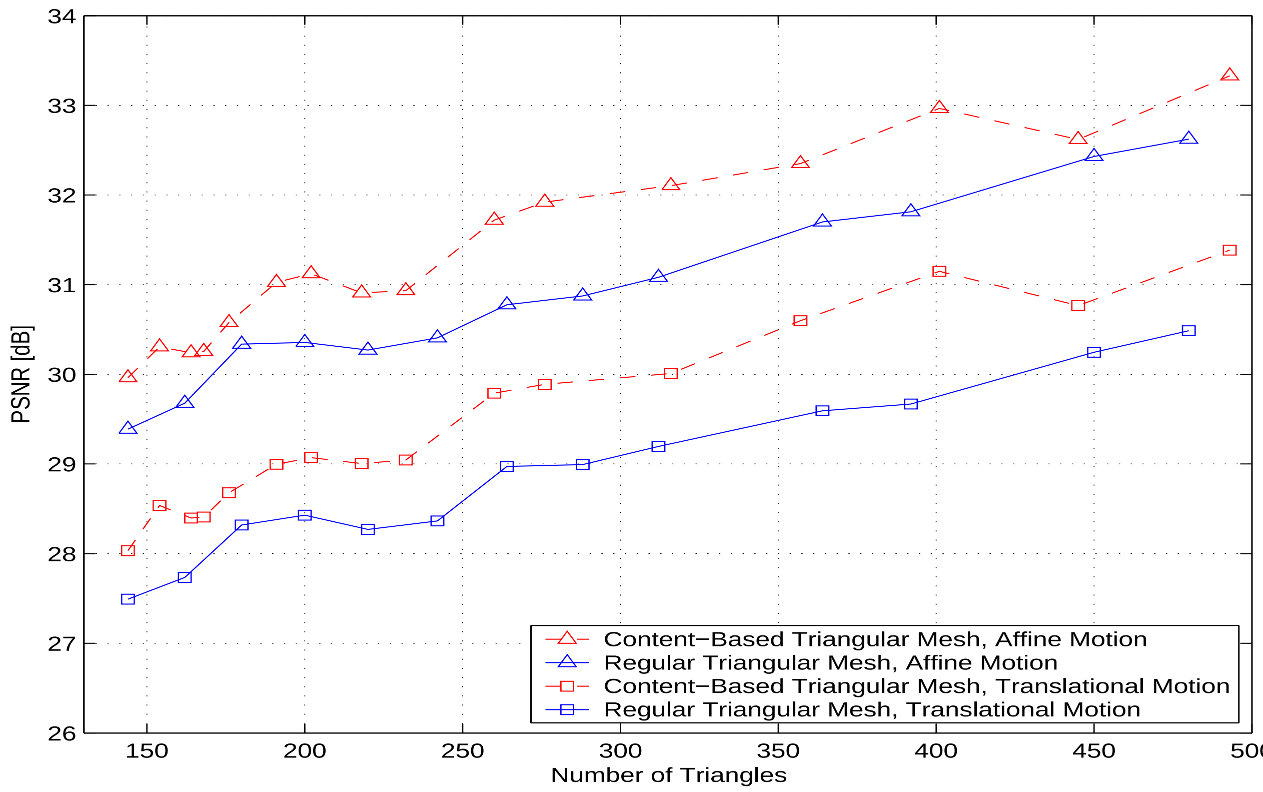


Figure 6: Luminance PSNR values of the motion compensated prediction error for frame 13 of the“Foreman” Sequence, motion compensated from frames 11 and 15. The effect of varying the number of triangles in the mesh is shown. The results for both content-based and regular meshes are provided.

objects. It is also evident that the use of an affine motion model allows for a more accurate representation of the actual motion within a region (triangle).

One frame from each sequence is shown in Figure 9 in order to illustrate the subjective quality of the methods used. In the case of the “Flower Garden” sequence, Figure 9(c) shows the regular triangular shaped artifacts which are particularly visible around the trunk of the tree. These are very similar to traditional blocking artifacts which occur when blocks straddle motion boundaries. These are less evident in Figure 9(e), where the content-based structure of the mesh has helped to preserve the trunk boundary.

For the “Football” sequence, triangular blocking-type artifacts are clearly visible in Figure 9(d), particularly in regions of fast motion. The use of a content-based mesh allows for a more accurate representation in Figure 9(f), although the polygon shape of certain objects is perhaps noticeable on closer inspection.

5 CONCLUSION

The results reported in this paper demonstrate that the use of a content-based triangular mesh for affine motion compensation offers significant advantages over traditional block-based methods. The resulting improvement in image quality was demonstrated both objectively and subjectively for three test sequences with relatively fast motion.

By using an affine model to approximate the motion within a region, motion can be represented more accurately than when using a purely translational model. It was shown that affine motion estimation can be performed reasonably efficiently using a (partial) search of $O(r^2)$. If used in a video codec, the affine motion parameters would need to be encoded, resulting in some additional overhead. For example, encoding the affine motion would require approximately eight bits per triangle (assuming a search radius of ± 3 pixels per vertex).

When using regular triangles (or blocks), a triangle can often cover an area occupied by two (or more) objects moving relative to one another. In this case, it is not possible to represent two different types of motion accurately with one set of (translation or affine) parameters. The method for designing a content-based mesh proposed in this paper attempts to prevent this problem by positioning triangles in such a way that they do not cross motion boundaries. Such a mesh typically has many small triangles in regions of significant motion, with much larger triangles covering regions of little or no motion.

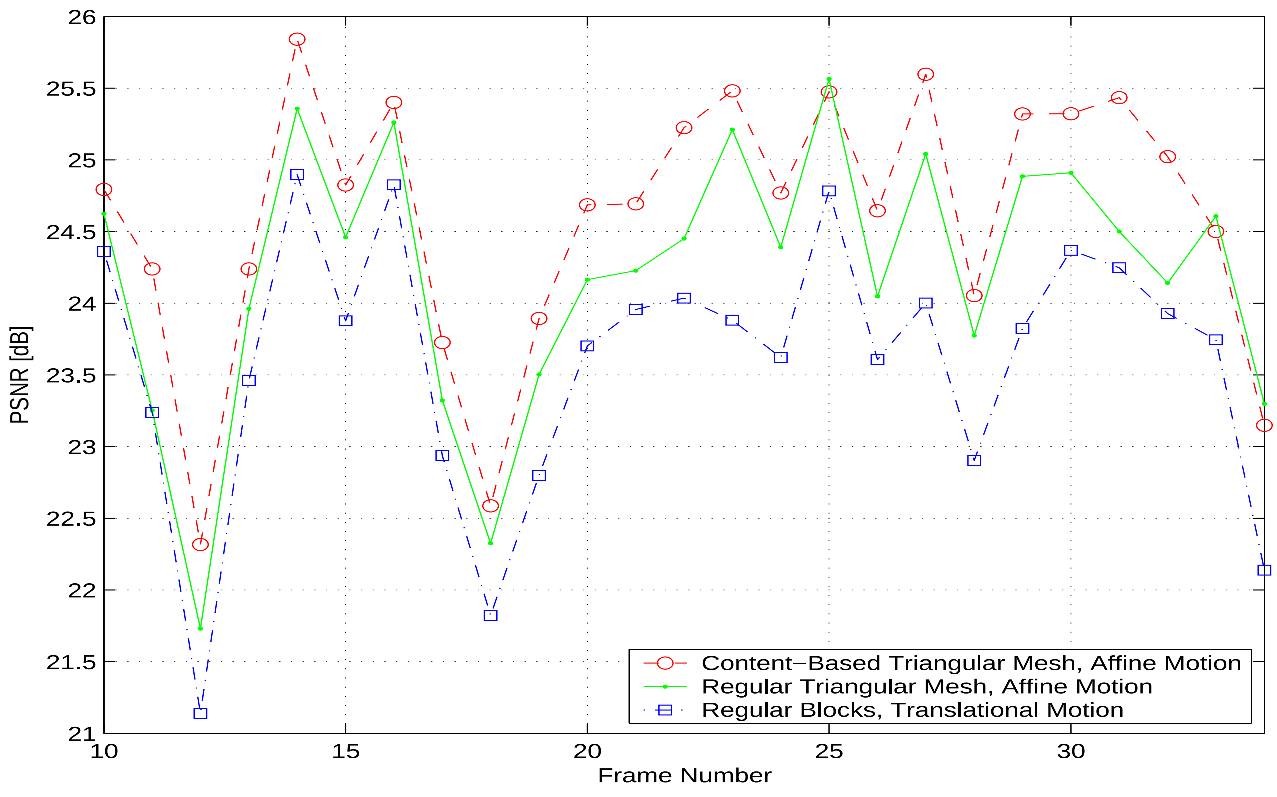


Figure 7: Luminance PSNR values of the motion compensated prediction error for frames 10 to 34 of the“Flower Garden” Sequence. Each (compensated) frame, $\tilde{I}_n$ has been motion compensated from the original frames I_{n-2} and I_{n+2}. On average 221 triangles (or blocks) per frame were used in the motion compensation process.

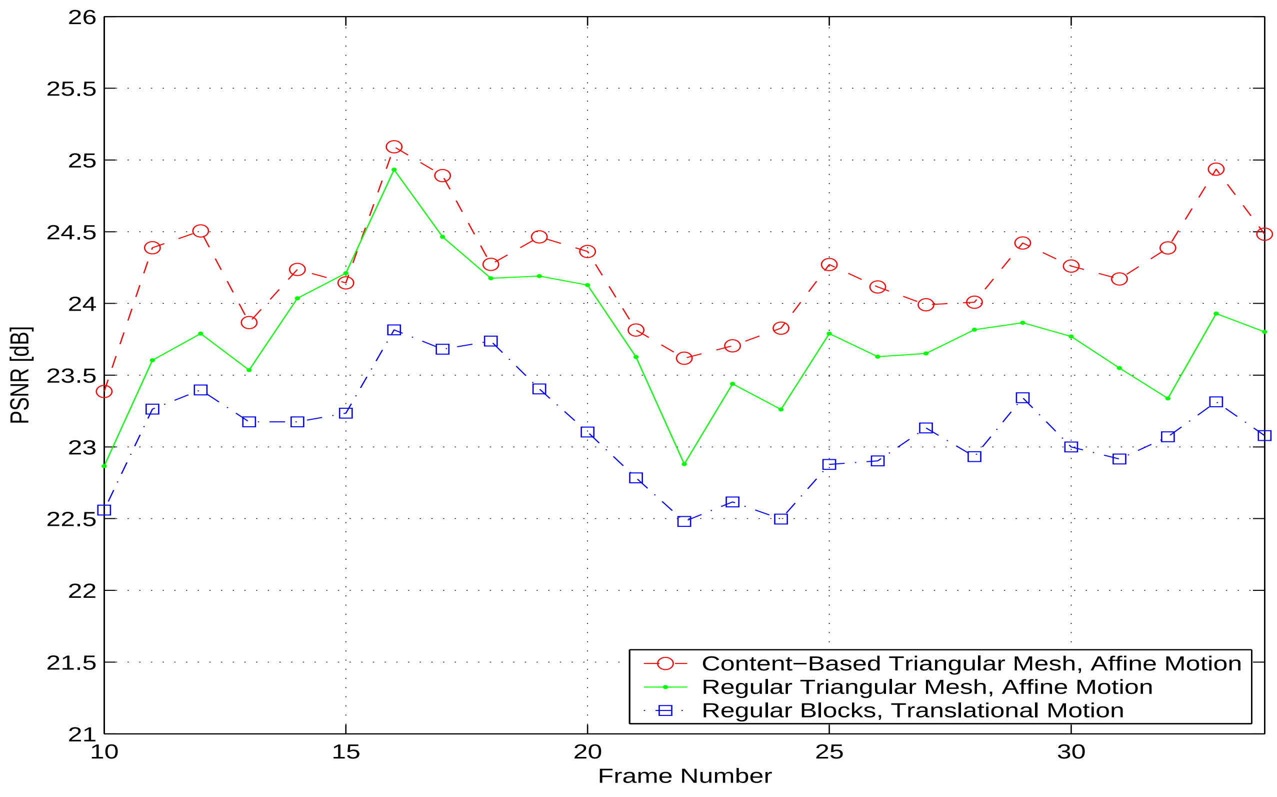
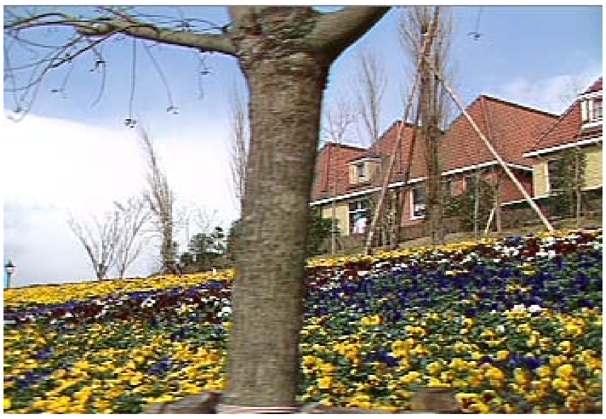
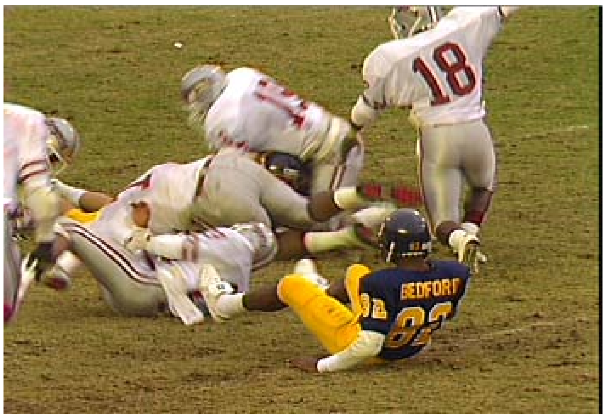
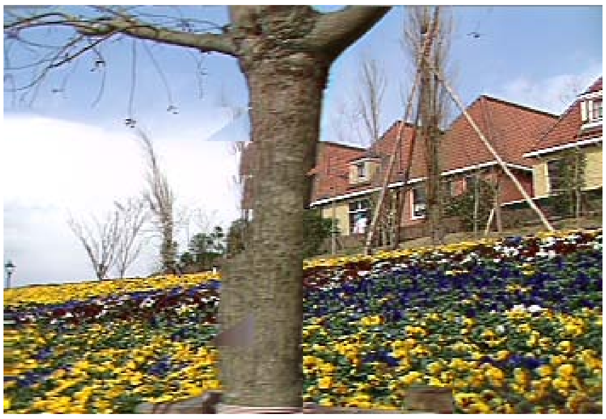
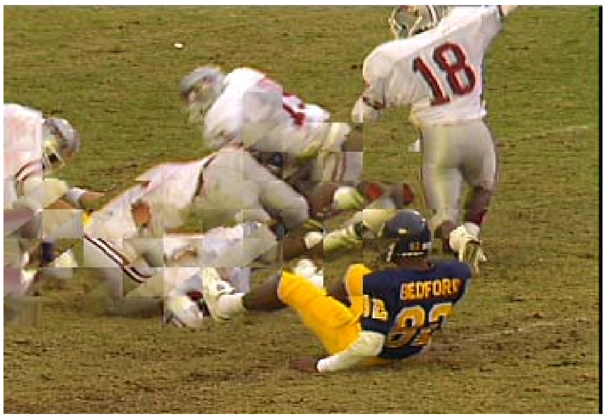


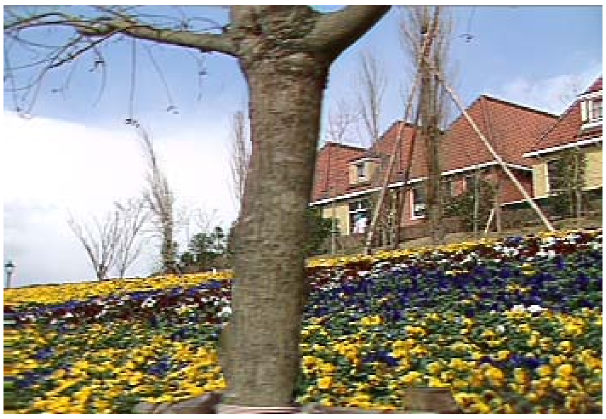
Figure 8: Luminance PSNR values of the motion compensated prediction error for frames 10 to 34 of the“Football” Sequence. Each (compensated) frame, $\tilde{I}_n$ has been motion compensated from the original frames I_{n-2} and I_{n+2}. On average 481 triangles (or blocks) per frame were used in the motion compensation process.


(a) Frame 12 (original) of “Flower Garden”


(b) Frame 32 (original) of “Football”


(c) Frame 12 motion compensated (from frames 10 and 14) using a regular triangular mesh with 288 triangles; PSNR = 25.36 dB


(d) Frame 32 motion compensated (from frames 30 and 34) using a regular triangular mesh with 420 triangles; PSNR = 23.34 dB


(e) Frame 12 motion compensated (from frames 10 and 14) using a content-based triangular mesh with 275 triangles; PSNR = 25.84 dB

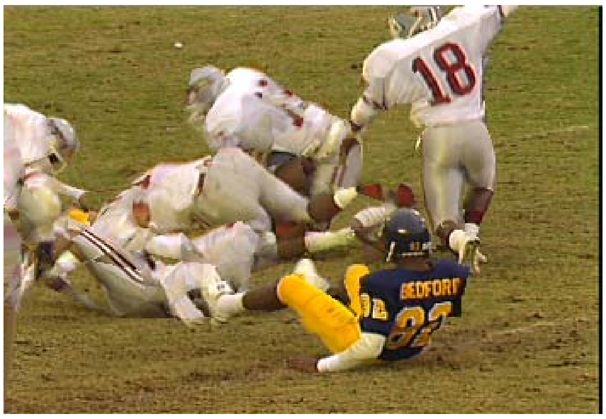

(f) Frame 32 motion compensated (from frames 30 and 34) using a content-based triangular mesh with 413 triangles; PSNR = 24.39. dB

Figure 9: Original and motion compensated frames from the “Flower Garden” and “Football” sequences (using affine motion compensation).

When considering its use in a practical video coding system, a content-based mesh poses one obvious problem: The mesh design process proposed in this paper would require the polygon boundaries to be known at the decoder,[6] resulting in a significant overhead. Methods for evolving the mesh from one frame to the next and/or coding the polygon boundaries efficiently are currently being researched. Initial results indicate that the cost of specifying the polygon boundaries is in the order of 20 bits per node.

The use of a non-connected mesh enables the motion parameters for each triangle to be calculated independently of other triangles, and thus allows for easy comparison with a regular mesh or a block-based approach. However, one idea currently being investigated is to use a connected mesh within each polygon-shaped region. This may lead to improved subjective quality by ensuring continuous motion within each object.

ACKNOWLEDGEMENTS

This work has been funded by BBC Research and Development and by the Centre for Vision, Speech and Signal Processing at the University of Surrey.

REFERENCES

[1] V. Seferidis and M. Ghanbari, 1993, General approach to Block-Matching Motion Estimation, Optical Engineering, 32:7, pp. 1464–1474.

[2] Y. Nakaya and H. Harashima, 1994, Motion Compensation based on Spatial Transforms, IEEE Trans. Circuits and Systems for Video Technology, 4:3, pp. 339–356.

[3] J. Nieweglowski, T. Campbell, and P. Haavisto, 1993, A Novel Video Coding Scheme based on Temporal Prediction using Digital Image Warping, IEEE Trans. Consumer Electronics, 39:3, pp. 141–150.

[4] M. Dudon, O. Avaro, and C. Roux, 1997, Triangular active mesh for motion estimation, Signal Processing: Image Communication, 10:1–3, pp. 21–41.

[5] Signal Processing: Image Communication, 15:4–5, January 2000, Tutorial Issue on the MPEG-4 Standard.

[6] D. B. Bradshaw and N. G. Kingsbury, 1997, A Combined Affine and Translational Motion Compensation Scheme using Triangular Tessellations, in Proc. Int. Con. on Acoustics, Speech, and Signal Processing (ICASSP), 2, pp. 2645–2648.

[7] D. B. Bradshaw, 1998, Motion Estimation and Compensation of Video Sequences using Affine Transforms, Ph.D. dissertation, Cambridge University, [Online]: http://www-sigproc.eng.cam.ac.uk/publications/theses.html

[8] Y. Altunbasak and A. M. Tekalp, 1997, Occlusion-Adaptive, Content-Based Mesh Design and Forward Tracking, IEEE Trans. Image Processing, 6:9, pp. 1270–1280.

[9] P. E. Eren and A. M. Tekalp, 2003, Bi-Directional 2-D Mesh Representation for Video Object Rendering, Editing and Superresolution in the Presence of Occlusion, Signal Processing: Image Communication, 18:5, pp. 313–336.

[10] T. Meier and K. N. Ngan, 1999, Video Segmentation for Content-Based Coding, IEEE Trans. Circuits and Systems for Video Technology, 9:8, pp. 1190–1203.

[11] M. Kim, J. G. Choi, D. Kim, H. Lee, M. H. Lee, C. Ahn, and Y.-S. Ho, 1999, A VOP Generation Tool: Automatic Segmentation of Moving Objects in Image Sequences Based on SpatioTemporal Information, IEEE Trans. Circuits and Systems for Video Technology, 9:8, pp. 1216–1226.

[12] P. Salembier and F. Marques, 1999, Region-Based Representations of Image and Video: Segmentation Tools for Multimedia Services, IEEE Trans. Circuits and Systems for Video Technology 9:8, pp. 1147–1169.

[13] Y. Deng, B. S. Manjunath, and H. Shin, 1999, Color image segmentation, Proc. IEEE Computer Society Conf. on Computer Vision and Pattern Recognition (CVPR), 2, pp. 446–451. [Online]: http://vision.ece.ucsb.edu/segmentation/jseg/

[14] M. J. Black and P. Anandan, 1996, The robust estimation of multiple motions: parametric and piecewise-smooth flow fields, Computer Vision and Image Understanding, 63:1, pp. 75–104. [Online]: http://www.cs.brown.edu/people/black/

[15] J. R. Shewchuk, 1996, Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator, Applied Computational Geometry: Towards Geometric Engineering, [Online]: http://www-2.cs.cmu.edu/ quake/triangle.html

[6]The triangles are generated for each polygon-shaped region using a prescribed algorithm.