A SIMPLEX-BASED PROJECTIVE TRANSFORM ESTIMATOR

J A Robinson

University of York, UK.

INTRODUCTION

The estimation of projective transforms between image pairs has important applications in computer vision. Some authors use pairwise projective transforms to stitch images together into mosaics [1][2] while others use them as the first stage in deriving depth and camera motion estimates [3][4]. More recent schemes apply global optimization, or at least adjustment, of transform estimates, but the pairwise estimation of projective transforms remains an important step. And while there are alternative transforms that can be used for mosaicing from a camera rotating about its optical centre [5][6], full projective estimation is necessary to mosaic a plane from a video sequence with free camera movement. In this paper I report a method for projective transform estimation between pairs of frames, and its use in a real-time mosaicing program.

The projective transform from an image in $[x, y]^{\mathrm{T}}$ to an image in $[\hat{x}, \hat{y}]^{\mathrm{T}}$ is given by

$$\begin{bmatrix} \hat{\mathbf{x}} \\ \hat{\mathbf{y}} \end{bmatrix} = \frac{\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}}{\begin{bmatrix} c_1 & c_2 \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} + 1} \qquad (1)$$

Estimation of the transform requires a search in eight-dimensional parameter space for a set $\{a_{11}, a_{12}, a_{21}, a_{22}, b_1, b_2, c_1, c_2\}$ that when applied to all the points in one image gives the closest possible match to the other image. Luminance values must be used to steer the search. A direct algorithm (like [1],[2] and the one reported here, as opposed to a feature-based algorithm [7]) works by minimizing some disparity measure between values in one image and values at corresponding transformed coordinates in the other. The usable domain and range of the transform depend on the image sizes and the transform parameters. Only the overlapping region of one image and a transformed version of the other can be used in a disparity calculation.

THE *SAM* ESTIMATOR

The estimator to be described here is called Simplex-Adapted Mesh (SAM). It has two stages – a translation estimator and a projective transform estimator. The first stage is conventional but described briefly below for completeness. The second stage involves the machinery referred to in the name – a sparse sample mesh and the Nelder-Mead simplex – and is more novel. For convenience I use *SAM* to refer to the whole method.

Stage 1: Reliable Translation Estimation

The estimator's first stage is a reliable translational estimate that provides tolerance to noise, illumination change, and small object movement within the scene. The estimator builds an image pyramid then does a wide area full-search match at the highest (i.e. smallest size) level. For every displacement up to half the image size, it determines the overlapping (matching) region and calculates the mean difference between the two images in this region. The mean is subtracted from each individual pel difference before summing up all the absolute values of differences in the overlapping region. The minimum mean disparity value provides the first estimate of translation at the highest level of the pyramid. This value is then projected down the pyramid, where it is refined level-by-level using a gradient descent algorithm. The difference in means of overlapping areas is also projected down and maintained in the search at each level. At the final (bottom) level of the pyramid, a full-resolution block the size of the overlapping area is used.

Because the translational stage begins with a wide area full search it matches accurately over a wide range. The continuous adjustment for the mean of the overlap region provides protection against global level change. This mean adjustment is carried forward to stage 2 of the estimator so that it too is resilient to changes in overall luminance level.

Stage 2: Simplex Optimization over a Mesh of Samples

The second stage of the estimator is a generate-and-test optimization procedure. It has two unusual features: first, its criterion function, and second, the optimization method.

SAM uses a mesh or grid of coordinates for which candidate transform values are calculated and which then sample the two images. A disparity value is computed from a weighted sum of the absolute differences in sample values. The weighting attenuates high absolute differences to limit the effect of localized moving objects.

The grid of samples may be square or quincunx. No prefiltering is used in either image, so at a local level, sample matches are subject to aliasing effects. But the mesh is not intended as a scaled image representation. Rather it is a way to distribute effectively random samplers over the candidate transformation's range and domain. The sample spacing is uniform – but that is simply a convenience for fast transform calculation. There is no relationship between image structure and where the sample mesh happens to fall, so clearly it is possible to be unlucky and miss useful image features. SAM relies on distributing enough samples through the image as a whole that this is unlikely to matter. In practice, all meshes finer than 1 in 16 image samples have comparable performance and SAM usually runs reliably with sparser meshes. The spacing is the only parameter of the method and varying from 1in 16 to 1 in 100 trades accuracy for speed.

The weighted sum of absolute differences over the sample mesh is used as the criterion function for optimization by the Nelder-Mead simplex method [8]. This method is not the linear programming simplex used in [9], but a general unconstrained optimization technique. It is the second unusual feature of the estimator. Its only other application to image correspondence analysis of which I am aware is [10], where it was used for estimating the translational motion of square blocks.

The Nelder-Mead simplex requires the maintenance of 9 candidate transforms and their iterative adjustment. Each candidate transform is a particular set of the eight parameters a_{11}, a_{12}, a_{21}, a_{22}, b_1, b_2, c_1, c_2, in equation 1 and therefore corresponds to a point in 8-dimensional parameter space. These nine points form the vertices of the simplex. Geometrically, the simplex method involves changing the shape of this hypersolid by systematic movement of the vertices towards the minimum, until they are close enough together to meet a termination condition.

The simplex method is sometimes regarded as inefficient because its exploration of directions in multidimensional space is guided by "accidental" configurations of simplex vertices rather than local gradient near the current search point. Yet this indeterminacy appears to work well in popping the simplex out of local minima. Because of the structure of pictures, local optimization minima occur near the global minimum. Once the "catchment area" of a local minimum has been reached by a gradient descent algorithm, that minimum will quickly be returned as the true minimum. In contrast, the simplex method continues to explore other possibilities as its vertices draw together. I have verified experimentally that using conjugate gradient descent in place of the simplex in SAM results in much more frequent convergence to false minima. I believe that this argument applies equally to other efficient local minimization schemes such as the version of Levenberg-Marquardt used in [1].

A second benefit of simplex is that the initialization of vertices can be done systematically, according to the expected variation in each of the dimensions. SAM does this by setting up the vertices one-by-one by orthogonal one-dimensional searches, with directions and step sizes derived from a prior analysis of many video samples. That is, the directions in which the simplex is initially constructed are chosen to shape it to explore the most common transforms encountered in practice. Each 1D search ends when the vertex and its predecessor span a good 1D minimum.

Finally, if the initialisation values are particularly bad for a given case, the simplex changes shape to move quickly towards better vertices. In doing so, it grows in size, automatically lengthening the search time, but ensuring that, once values in the vicinity of the minimum are found, the simplex will converge slowly enough to avoid false minima.

Once a minimum has been found it is possible to restart the simplex with noisy values as a check for a false minimum. In a large number of tests this has yielded an improved result in less than 0.5% of cases. The standard method therefore runs the simplex only once.

REAL-TIME MOSAICING PROGRAM

SAM is incorporated into a program for real-time mosaicing. The design criteria for this program were:

1. It must be fast enough to mosaic 320x240 pel frames at 5 frames/s.
2. It must be causal, that is, it can only use information in the past to estimate the transformation for the current frame.
3. When the system cannot find a good estimate, it must signal this to the user.
4. The user must be able to correct mistakes simply.

These criteria were fulfilled as follows:

1. SAM is used to achieve fast projective transform estimation. The program then transforms the current frame and pastes it

directly onto the current "forward" mosaic (with no sophisticated stitching mechanism to hide image boundaries).
2. The system does not attempt global registration of frames. Instead the current "forward" mosaic is inverse transformed to a "backward" mosaic centred on the previous frame, and this accumulation of all previous frames is used for transform estimation of the current frame.
3. When the system cannot find a good estimate, it displays the current input frame in monochrome and does not add it to the mosaic. When an input frame yields a reliable transform estimate, the program goes back to accumulating mosaics.
4. The system is self-correcting when the user revisits areas of error. New frames simply overwrite the erroneous part of the mosaic.

The real-time mosaicing program is described in more detail in [11].

EXPERIMENTS

I show representative examples of the operation of the mosaicing program incorporating SAM. The mesh spacing in all cases was 1 in 16. Many things can be done to improve the appearance of a mosaic, even after accurate estimation of the projective transform and correction for radial distortion. These include:

- Only using the central parts of each image, or weighting the central parts more heavily than the peripheries [1].
- Local adjustments to compensate for small parallax and radial distortions [12].
- Joining images in the mosaic along paths of least difference [7].

Any of these could be applied to re-stitch the images in post-processing, but none are used in the real-time program, and none are applied to the results shown in figures here.

Figure 1 shows every fifth frame from a webcam video sequence. Figure 2 shows an intermediate "backward" mosaic as displayed to the user during the capture of the sequence and figure 3 gives the final mosaic. Figure 4 shows frames from a sequence with a moving person and figure 5 the mosaic constructed from it with the person removed from all but his last position. Figure 6 is a mosaic derived from a video where the camera was moved freely in front of a plane.

Figure 7 is a real-time mosaic constructed from a camcorder sequence used by Davis [7]. This is a difficult sequence because the focal length and gain vary. The reader is invited to compare the SAM result in figure 7 with the pairwise Davis algorithm [13]. Davis's final result, achieved with global registration and least difference path stitching illustrates what may be achieved with post-processing [14]. Videos of the sequences associated with figures 1 to 6 and others are available online [15].

A series of experiments conducted with Li-Te Cheng compared the accuracy, robustness and speed of SAM against several alternatives. Only one of these was a full projective transform estimator [2], and in these comparisons SAM proved to be more reliable and thirty times faster than the alternative. Quantitative comparisons against other projective transform estimators are in progress.

REFERENCES

[1] R Szeliski, "Image Mosaicing for Tele-Reality Applications", Digital Equipment Corporation Cambridge Research Laboratory, Technical Report CRL 94/2, May 1994.
[2] S Mann, R Picard, "Video Orbits of the Projective Group: A simple approach to featureless estimation of parameters", IEEE Trans Image Proc, Vol 6, No 9, 1997, pp 1281-1295.
[3] R Kumar, P Anandan, K Hanna, "Shape Recovery from Multiple Views: A Parallax Based Approach", Proc 12th Internat Conference on Pattern Recognition, 1994, pp685-688.
[4] M Irani, B Rousso, S Peleg, "Recovery of Ego-Motion using Region Alignment", IEEE Trans PAMI, Vol 19, No 3, 1997, pp 268-272.
[5] S Peleg, J Herman, "Panoramic Mosaics with VideoBrush, "DARPA Image Understanding Workshop, May 1997, pp 261-264.
[6] J Davis, "Mosaics of Scenes with Moving Objects", Proc CVPR'98, June 1998.
[7] P H S Torr, A Zisserman, "Feature Based Methods for Structure and Motion Estimation", and following discussion, in B Triggs, A Zisserman, R Szeliski (eds.), "Vision Algorithms: Theory and Practice. Proc Internat Workshop on Vision Algorithms, Corfu, Sept 1999, Springer-Verlag Lecture Notes in Computer Science, 1883, pp 267-297.
[8] J A Nelder, R Mead, "A Simplex Method for Function Minimization", The Computer Journal, Vol 7, 1965, pp 308-313.
[9] M Ben-Ezra, S Peleg, M Werman, "Real-Time Motion Analysis with Linear Programming", CVIU, Vol 78, No 1, pp 32-52, 2000.
[10] M E Al-Mualla, C N Canagarajah, D R Bull, "Simplex Minimization for Single- and Multi-Reference Motion Estimation", IEEE Trans on Circuits and Systems for Video Tech, Vol 11, No 12, Dec 2001, pp 1209-1220.

[11] J A Robinson, “Collaborative Vision and Interactive Mosaicing”, submitted to the First Internat Conference on Vision, Video and Graphics, July 2003.
[12] H-Y Shum, R Szeliski, “Panoramic Image Mosaics”, Microsoft Research Technical Report, MSR-TR-97-23, 1997.
[13] http://graphics.stanford.edu/~jedavis/panorama/images/memchu.proj.jpg
[14] http://graphics.stanford.edu/~jedavis/panorama/images/memchu.w.lg.jpg
[15] http://www.elec.york.ac.uk/visual/mosaicing/

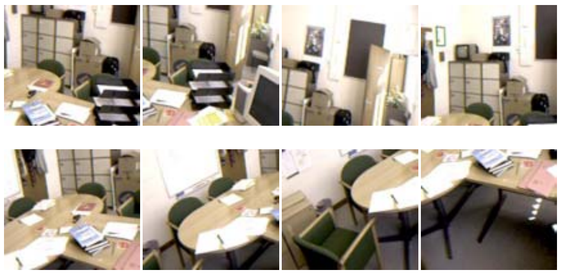
Fig 1: Frames from an example input webcam video including revisits.

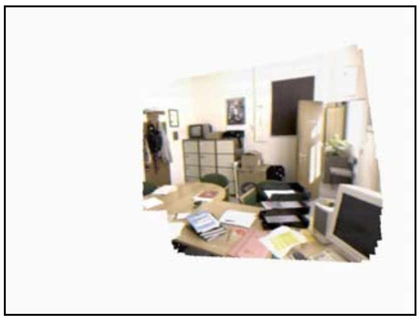
Fig 2: Early intermediate mosaic from the video in figure 1

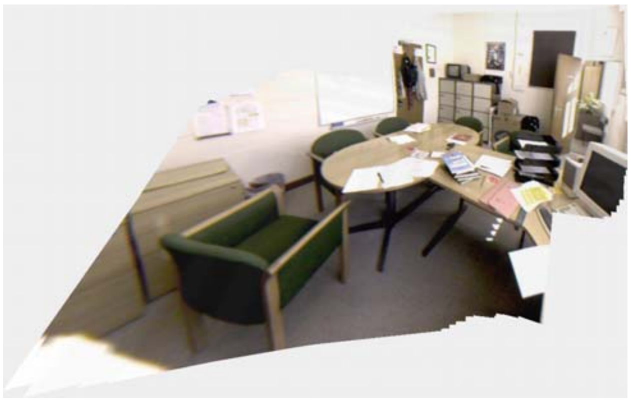
Fig 3: Final mosaic from the frames in fig 1.

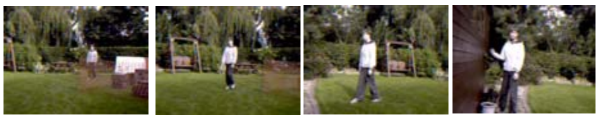
Fig 4: Frames from an example input webcam video including a moving person.

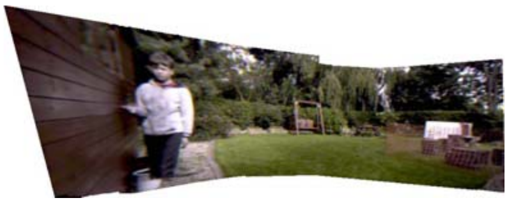
Fig 5: Final mosaic from the frames in fig 4.

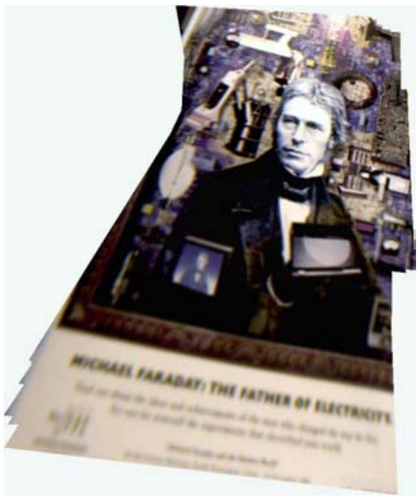

Fig 6: Mosaic generated by free movement of the camera over a plane. The image is preserved at higher resolution than shown in this particular view, where the mosaic is centred on the final video frame, viewing Faraday’s head from an upward-tilted camera. After mosaicing, the plane can be reoriented as desired.

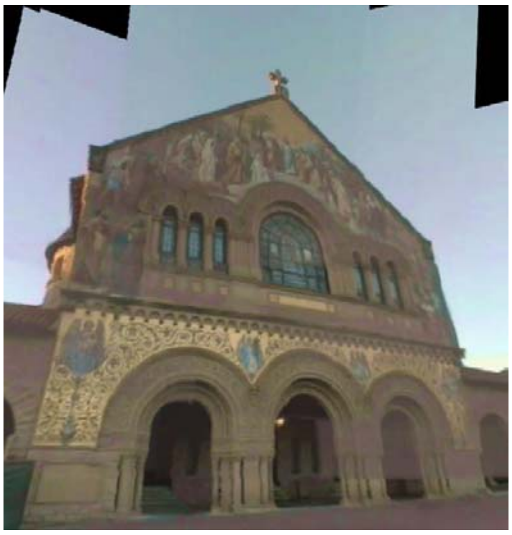
Fig 7: Mosaic generated from Davis’ “memchu” sequence. The streaks in the bottom right occur because variation in camera gain is not compensated for. However, use of SAM on successive frames of a 150-frame sequence has allowed reasonable recovery of image geometry in a single pass with no global correction.