MULTI-RESOLUTION PARAMETRIC REGION TRACKING FOR 2D OBJECT REPLACEMENT IN VIDEO

Paul Brasnett, David R. Bull & Nishan Canagarajah

Dept. of Electrical and Electronic Engineering
University of Bristol
paul.brasnett@bristol.ac.uk

ABSTRACT

This paper develops an efficient parametric multi-resolution region tracking algorithm which is applied to the task of object replacement in video sequences. The tracker relies on gradient-based techniques to provide efficient estimation of the target location and pose in each frame. The tracking algorithm improves upon similar efficient parametric algorithms by increasing the distance an object can move from frame to frame. Experimental results provide clear evidence of the improved performance over existing region tracking.

The parameters that estimate the pose and location of the target are used to initialise the object replacement algorithm. The object replacement uses the pose estimates with texture mapping techniques to perform image warping of an arbitrary sized *replacement* image. The location estimates are used to accurately insert the replacement image into the current frame.

1. INTRODUCTION

We introduce a method for the efficient (real-time) tracking and replacement of a planar surface in a video sequence. To enable the replacement of the surface it is necessary to use a parametric motion model for the tracking task. These parameters can then be used to correctly introduce the new surface.

The field of work referred to as augmented reality concerns the combining of computer generated images (CGI) with real world images. Work in this area tends to concentrate on inserting CGI objects into real world video [1] whilst some look at both 2D and 3D object replacement [2]. This paper looks at the task of surface replacement.

Common approaches to tracking include template matching in both spatial and frequency domains, differential techniques and energy based methods. In comparison gradient-based methods have been found to perform well [3].

Moving objects and/or a moving camera can cause changes to the apparent geometry of objects throughout a video sequence. Methods for tracking these changes include sum-of-squared difference (SSD) based approaches [4]. These approaches work satisfactorily when the motion model is simple or of low dimension such as pure translation but become computationally complex with higher-dimensional models. Alternatively, if frequency domain techniques are used, they often fail to cope with higher dimensional models because of a lack of scale invariance. To allow accurate object replacement it is necessary to have parametric estimates for the motion, so scale invariance is not a viable solution here.

Some efficient SSD-based methods make use of eigenspace representation whilst other techniques make use of gradient information. In [5] an eigenspace representation is combined with Levenburg-Marquandt optimisation techniques whilst [6] uses the reference template gradients with a Newton-Raphson style optimisation. Both of these algorithms are efficient but both approaches can only track the target over small displacements.

The authors extend the work of [6] to provide the ability to track faster object motion through the introduction of a multi-resolution approach [7, 8]. The multi-resolution representation used is the Gaussian pyramid . The output from the algorithm is then used for texture mapping to replace target regions in a video sequence.

The paper first explains the efficient gradient-based approach to region tracking in Section 2. Section 3 introduces the multi-resolution extension to the efficient gradient-based approach developed by the authors. Section 4 describes the process of the parameter estimation and the application of the parameters to the texture mapping of the replacement object. Results are presented in Section 5 and conclusions are presented in Section 6.

2. EFFICIENT GRADIENT-BASED REGION TRACKING

We can denote the intensity, $I(\boldsymbol{x}, t)$, at the location $\boldsymbol{x} = (x, y)^t$ in an image acquired at time t. We can then define the spatial gradient at that location and time as $\nabla_x I(\boldsymbol{x}, t)$. We can model the relative motion which occurs between the camera and the target object using a parametric motion model $\boldsymbol{f}(\boldsymbol{x}; \boldsymbol{\mu})$ where the motion parameter vector $\boldsymbol{\mu}$

0-7803-8554-3/04/$20.00 ©2004 IEEE.

is given by $\boldsymbol{\mu} = (\mu_1, \mu_2, \ldots, \mu_n)$ and $\boldsymbol{f}$ is differentiable in both $\boldsymbol{\mu}$ and $\boldsymbol{x}$. The task of tracking an object involves the computing of $\boldsymbol{\mu}$ at time t. The parameters for the motion model in the reference frame are denoted $\boldsymbol{\mu}_0$. A *target region* is defined as the set of N image locations $\mathcal{R} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N)^T$. The first frame in the sequence is defined as the *reference image* and the *reference template* can be described as $\boldsymbol{I}(\boldsymbol{\mu}_0, t_0)$ if we define

$$\boldsymbol{I}(\boldsymbol{\mu}, t) = (I(\boldsymbol{f}(\boldsymbol{x}_1, \boldsymbol{\mu}), t), \ldots, I(\boldsymbol{f}(\boldsymbol{x}_N, \boldsymbol{\mu}), t))^T \quad (1)$$

as the brightness values of the target region at time t.

If we make use of the *image constancy assumption* [9] and assume that the motion model completely describes changes to the target region then tracking becomes the minimisation of the difference between the reference target and the target region in the current frame. This is expressed in the following least squares objective function

$$O(\boldsymbol{\mu}) = \sum_{x \epsilon \mathcal{R}} (I(\boldsymbol{f}(\boldsymbol{x}; \boldsymbol{\mu}), t) - I(\boldsymbol{x}, t_0))^2 \quad (2)$$

This can be reworked in to the task of minimising the objective function

$$O(\delta\boldsymbol{\mu}) = \|\boldsymbol{I}(\boldsymbol{\mu}(t) + \delta\boldsymbol{\mu}, t + \tau) - \boldsymbol{I}(\boldsymbol{\mu}_0, t_0)\|^2 \quad (3)$$

Solving the set of equations $\nabla O = 0$ yields the solution

$$\delta\boldsymbol{\mu} = -(\boldsymbol{M}^t\boldsymbol{M})^{-1}\boldsymbol{M}^t[\boldsymbol{I}(\boldsymbol{\mu}, t + \tau) - \boldsymbol{I}(\boldsymbol{\mu}_0, t_0)] \quad (4)$$

Where $\boldsymbol{M}$ is the Jacobian matrix of the target region to be tracked. This result holds provided that the matrix $\boldsymbol{M}^T\boldsymbol{M}$ evaluated at $(\boldsymbol{\mu}, t)$ has full rank. We can further define an error vector

$$\boldsymbol{e}(t + \tau) = \boldsymbol{I}(\boldsymbol{\mu}(t), t + \tau) - \boldsymbol{I}(\boldsymbol{\mu}_0, t_0) \quad (5)$$

Combining Equations (4) and (5) we obtain an expression for the change in motion parameters from the previous frame

$$\delta\boldsymbol{\mu} = -(\boldsymbol{M}^T\boldsymbol{M})^{-1}\boldsymbol{M}^T\boldsymbol{e}(t + \tau) \quad (6)$$

Equation (6) can be restated as the new motion parameters, as in Equation (7)

$$\boldsymbol{\mu}(t + \tau) = \boldsymbol{\mu}(t) - (\boldsymbol{M}^T\boldsymbol{M})^{-1}\boldsymbol{M}^T\boldsymbol{e}(t + \tau) \quad (7)$$

To implement an efficient algorithm the online computation can be reduced with the following substitution

$$\boldsymbol{M}(\boldsymbol{\mu}) = \begin{bmatrix} \nabla_{\boldsymbol{x}} I(\boldsymbol{x}_1, t_0)^T \boldsymbol{\Gamma}(\boldsymbol{x}_1) \\ \nabla_{\boldsymbol{x}} I(\boldsymbol{x}_2, t_0)^T \boldsymbol{\Gamma}(\boldsymbol{x}_2) \\ \vdots \\ \nabla_{\boldsymbol{x}} I(\boldsymbol{x}_N, t_0)^T \boldsymbol{\Gamma}(\boldsymbol{x}_N) \end{bmatrix} \boldsymbol{\Sigma}(\boldsymbol{\mu}) = \boldsymbol{M}_0 \boldsymbol{\Sigma}(\boldsymbol{\mu}) \quad (8)$$

Where $\boldsymbol{M}_0$ is the Jacobian matrix of the target region in the reference frame and $\boldsymbol{\Gamma}$ is dependent upon the motion model, we describe the details for the affine motion model in Section 3.2.

3. EFFICIENT MULTI-RESOLUTION REGION TRACKING

It is desirable to combine the improved range of motion provided by the lower resolution processing with the higher accuracy of the higher resolution processing. These requirements lead to the following proposal for a multi-resolution algorithm based on the efficient algorithm described above. The initial track occurs at a low resolution and then the results are refined at higher resolutions until the desired accuracy is achieved. We make use of a *Gaussian pyramid* as the structure for this coarse-to-fine processing.

3.1. Tracking Algorithm

As part of the efficient implementation the algorithm is split into offline and online algorithms, both described below.

Algorithm: Offline Stage

1 Obtain target region from the reference frame;
2 Construct the Gaussian pyramid for the reference frame, with the level in the pyramid defined by l, where $l <= L$;
3 Compute $\boldsymbol{M}_{0,l}$ and $\boldsymbol{\Lambda}_l = \boldsymbol{M}_{0,l}^t \boldsymbol{M}_{0,l}$;

Algorithm: Online Stage

foreach *Frame* ϵ *Sequence* **do**
1 Generate a Gaussian pyramid for the current frame;
foreach $l <= L$ **do**
2 Rectify the current frame using the motion parameters $\mu_l^c(t)$;
3 Compute $e_l(t + \tau)$;
4 Solve $\Sigma_l^T \Lambda_l \Sigma_l = \Sigma_l^T M_{0,l} e_l(t + \tau)$ for $\delta\mu_l$ where Σ_l is evaluated at $\mu_l^c(t)$;
5 $\mu_l(t + \tau) = \mu_l^c + \delta\mu_l$;
if $l = 1$ **then**
6 Compute $\mu_L^c(t + \tau)$;
else
7 Compute $\mu_{l-1}^c(t)$;
end
end
8 $\mu(t + \tau) = \mu_1(t + \tau)$;
end

During processing of the Gaussian pyramid in the online stage (steps 2-7) we proceed from the lowest level (L) up to the highest level (1). In Step 6 each new frame is initialised with the *scale corrected* parameters from the previous frame, $\mu_L^c(t + \tau)$. Step 7 *scale corrects* the motion parameters for propagation through the levels of the pyramid.

3.2. Affine Motion Model

The implementation of the algorithm proposed here makes use of an affine transform for the motion model. It is possible to implement other linear models such as pure translation or translation, rotation and scale as well as some non-linear motion models that can be expressed as either $y = f(x)$ or $x = g(y)$[6]. The affine motion model can be expressed as

$$\boldsymbol{f}(\boldsymbol{x};\boldsymbol{\mu}_l) = \begin{bmatrix} a & c \\ b & d \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} u_l \\ v_l \end{bmatrix} = \boldsymbol{A_l x} + \boldsymbol{u}_l \qquad (9)$$

Where parameters of the motion model are given by $\boldsymbol{\mu}_l = (u_l, v_l, a, b, c, d)^T$. From this we can obtain

$$\boldsymbol{\Gamma}(\boldsymbol{x}) = \begin{bmatrix} 1 & 0 & x & 0 & y & 0 \\ 0 & 1 & 0 & x & 0 & y \end{bmatrix} \qquad (10)$$

and

$$\boldsymbol{\Sigma}(\boldsymbol{\mu}_l) = \begin{bmatrix} \boldsymbol{A}^{-1} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{A}^{-1} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{A}^{-1} \end{bmatrix} \qquad (11)$$

Since $\boldsymbol{\Sigma}$ is invertible we can express the solution to the linear system as

$$\delta\boldsymbol{\mu}_l = -(\boldsymbol{\Sigma}_l^{-1})^T (\boldsymbol{M}_{0,l}^T \boldsymbol{M}_{0,l})^{-1} \boldsymbol{M}_{0,l}^T \boldsymbol{e}(t+\tau) \qquad (12)$$

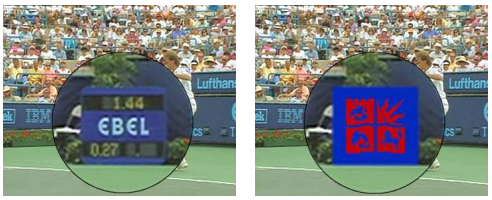
We can therefore compute the factor $(\boldsymbol{M}_{0,l}^T \boldsymbol{M}_{0,l})^{-1} \boldsymbol{M}_{0,l}^T$ offline thereby reducing the online computation. This leaves the online computation at each level as n N-vector inner products and n n-vector inner products. The Gaussian pyramid representation means that the number of pixels in the target region, N (the number of pixels), is reduced by a factor of four in each extra layer we introduce. So, for the multi-resolution tracking algorithm this tends to $1\frac{1}{3}n$ N-vector inner products and Ln n-vector inner products.

3.3. Track Quality

A measure of *tracking quality* is introduced to monitor the performance of the tracker. The quality is defined as the normalised SSD residual for the frame at the highest resolution in the Gaussian pyramid. If this residual is above a threshold, a further iteration is started with the output motion parameters passed back into the lowest level of the pyramid. If the residual has not dropped below the threshold after 3 iterations then the object is declared *lost*.

4. SURFACE REPLACEMENT

After tracking we replace the tracked object with the replacement image (which is initially arbitrarily sized). The required result is exemplified by Figure 1. The offline stage here involves resizing the replacement region to the size of the target region in the reference frame. This is done using bi-cubic interpolation.

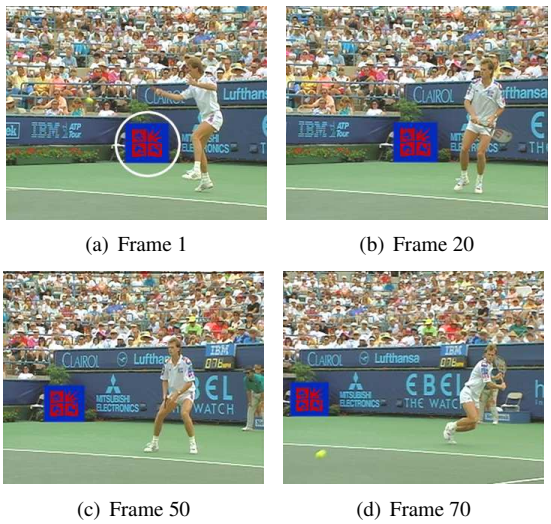


(a) Original Frame (b) Augmented Frame

Fig. 1. A frame from the *Stefan* sequence showing the frame before and after the augmentation processes

Once the target region in the current frame is located we use texture mapping techniques to perform image warping on the replacement region using the pose parameters obtained from the tracking algorithm. A low-pass filter applied to the edge of the inserted region improves the visual appearance.

5. RESULTS



(a) Frame 1 (b) Frame 20

(c) Frame 50 (d) Frame 70

Fig. 2. A section from the *Stefan* sequence with the score board tracked and replaced

The results of testing the tracker with various sequences shows promising performance. The general non-iterative scheme is able to track relative target motion between frames of up to about 8 pixels per frame. Whilst the iterating (Section 3.3) scheme is able to cope with relative target motion of around 10 pixels per frame. Results for tests of the combined tracking-replacement on the Stefan sequence can be seen in Figure 2. The target region in the last frame has undergone a motion of 10 pixels from the previous frame.

The accuracy of the tracker is critical to the performance of the combined scheme so we investigate the tracking ac-

Table 1. Results of object tracking with different numbers of levels in the Gaussian pyramid

Sequence	Pyramid Levels				Target Region	
	1	2	3	4	Size	Total Pixels
1	⊙				60×54	3240
2	⊙	⊙			61×55	3355
3	×				60×54	3240
4	×	⊙			86×55	4730
5	×	×		×	54×33	1782
6	⊙	⊙	⊙	×	41×42	1722
7	×	⊙		⊙	53×33	1749
8	×	⊙		×	44×30	1320
9	⊙				46×51	2346

×=Failed Tracking, ⊙=Partially Successful Tracking

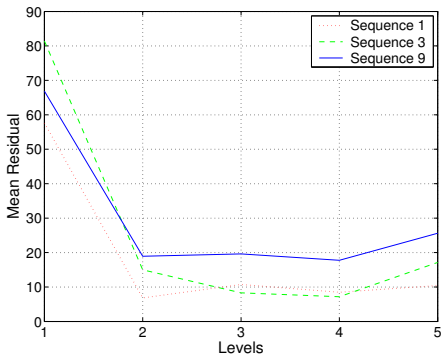


Fig. 3. Graph comparing the performance of the algorithm with different numbers of levels in the Gaussian pyramid

curacy further. The results from nine different sequences of varying lengths between 144 frames and 220 frames are shown in Table 1. The tests were carried out to investigate the tracker's behavior when a different number of levels are used in the Gaussian pyramid. The table shows when the algorithms fail to track the object (×), when the algorithm tracks successfully for part of the sequence (⊙) and all others are successful tracks. Having one level in the pyramid is equivalent to the original single resolution algorithm [6]. It can be seen that without the multi-resolution approach the tracker fails to track through a complete sequence successfully. Whereas with three levels in the pyramid the objects are tracked successfully in all but one sequence. Sequences 5-7 show that we can not indefinitely add more levels. The reason behind this is that down-sampling by a factor of two repetitively leads to a target region that is too small to provide gradient information and the higher levels are unable to correct the error introduced at the lower resolution.

We use the mean SSD residual to investigate the performance of the algorithm in more detail through sequences 1,3 and 9, the residuals can be seen in Figure 3. We see that the residual for one pyramid level is very high, this is due to the partial failing of the tracker when the target moves too quickly and the tracker getting caught in a local minimum. However, we also see that two, three and four levels give comparable performance. With five levels the performance again starts to deteriorate because the target region has become too small. Comparing sequences 1 and 3 with sequence 9 we can see that the performance is better with larger target regions.

6. CONCLUSION & FUTURE WORK

We have developed here an efficient algorithm for tracking objects in video sequences and have then applied this to the task of surface replacement using texture mapping. The results in Section 5 are evidence for the improvements provided by the authors' multi-resolution approach compared to similar algorithms.

Further investigation could include the benefits of different motion models, occlusion handling and extraction of illumination parameters in order to improve the appearance of the replaced object and the benefit of dynamically adapting the levels depending on the speed of object motion.

7. REFERENCES

[1] K. Cornelis, M. Pollefeys, M. Vergauwen, and L. Van Gool, "Augmented reality from uncalibrated video sequences," in *3D Structure from Images - SMILE 2000: Second European Workshop on 3D Structure from Multiple Images of Large-Scale Environments*, 2001, vol. 2018 / 2001, pp. 144–160.

[2] M. Jethwa, A. Zisserman, and A. Fitzgibbon, "Real-time panoramic mosaics and augmented reality," 1998.

[3] J.L. Barron, D.J. Fleet, S.S. Beauchemin, and T.A. Burkitt, "Performance of optical flow techniques," *CVPR*, vol. 92, pp. 236–242.

[4] J. Lewis, "Fast normalized cross-correlation," in *In Vision Interface*, 1995.

[5] Michael J. Black and Allan D. Jepson, "Eigentracking: Robust matching and tracking of articulated objects using a view-based representation," in *ECCV (1)*, 1996, pp. 329–342.

[6] Gregory D. Hager and Peter N. Belhumeur, "Efficient region tracking with parametric models of geometry and illumination," *IEEE Trans. on PAMI*, vol. 20, no. 10, pp. 1025–1039, 1998.

[7] R. Szelski and J. Coughlan, "Hierarchical spline-based image registration," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 94)*, June 1994, pp. 194–201.

[8] E. P. Simoncelli, "Coarse-to-fine estimation of visual motion," in *Proc Eighth Workshop on Image and Multidimensional Signal Processing*, 1993, pp. 128–129.

[9] B.K.P. Horn, *Robot Vision*, MIT Press, 1986.