Local Application of Optic Flow to Analyse Rigid versus Non-Rigid Motion

Alan J. Lipton
The Robotics Institute, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA, 15213
email: ajl@cs.cmu.edu, URL: http://www.cs.cmu.edu/˜ vsam

Abstract

Optic flow has been a research topic of interest for many years. It has, until recently, been largely inapplicable to real-time video applications due to its computationally expensive nature. This paper presents a new, reliable flow technique called dynamic region matching*, based on the work of Anandan[1], Lucas and Kanade[10] and Okutomi and Kanade[11], which can be combined with a motion detection algorithm (from stationary or stabilised camera image streams) to allow flow-based analyses of moving entities in real-time. If flow vectors need only be calculated for "moving" pixels, then the computation time is greatly reduced, making it applicable to real-time implementation on modest computational platforms (such as standard Pentium II based PCs).*

Applying this flow technique to moving entities provides some straightforward primitives for analysing the motion of those objects. Specifically, in this paper, methods are presented for: analysing rigidity and cyclic motion using residual flow*; and determining self-occlusion and disambiguating multiple, mutually occluding entities using* pixel contention.

Keywords: optic flow, motion analysis, tracking

1 Introduction

Analysing the motion of entities in a video stream is an important, current research challenge. Groups such as the entertainment industry use motion captured from video imagery to generate computer graphic characters and avatars for movies, computer games, and web-based applications. These motion capture techniques usually require a large investment of operator time and effort. As applications become more interactive, it will be increasingly important to automate the analysis of moving objects in real time. In applications such as automated video surveillance[7], it is essential to be able to understand the different motions of objects and infer their behaviours - such as the difference between two joggers meeting in a park, and a mugging.

Presented in this paper is a new strategy for divining some low-level motion parameters in real-time from entities in video streams. A new optic flow technique called *dynamic region matching* is applied locally to *blobs* extracted using an adaptive background subtraction motion detection algorithm[4]. Computing flow only for "moving" pixels allows real-time implementation on even modest computational platforms (such as Pentium II PCs). The resulting local flow fields can be used to analyse the motions of those *blobs*. Specifically, this technique is used to determine rigidity of motion, determine self-occlusion, and disambiguate mutually occluding moving objects.

1.1 Optic Flow

Optic flow techniques are traditionally used to determine camera motion or reconstruct 3D scene structure. In these cases, only the gross flow of large scene components is required and real-time performance is not necessary. Consequently, most flow techniques provide sparse results over an entire image or are extremely computationally expensive. Furthermore, most algorithms fail in largely homogeneous regions (ie. regions lacking texture). Although Anandan[1] provides a measure for determining just how bad a flow vector is, most algorithms make no effort to improve flow vectors in homogeneous regions.

Barron, Fleet, and Beauchemin[2] divide flow techniques into four categories: differential methods; region-based matching; energy-based; and phase-based. Of these, the most amenable to real-time implementation are the differential methods and the region-based matching methods. The differential methods such as Lucas and Kanade[10] effectively track intensity gradients in the scene. The advantage of this is that reliable flow vectors can be determined based on the *information* (measured by intensity gradient) content of a scene. However, one of the assumptions behind this method is that there is no deformation in the objects in the scene. Clearly an unreasonable assumption when determining the motion of a entity like a human being. Region-based matching techniques such as Anandan[1] track small regions in the scene from frame to frame. These methods are more robust to deformable objects, but are susceptible to large errors where there is little texture in the scene.

1.2 Motion Analysis

The ultimate goal of motion analysis is to describe activities occurring in video streams. Primitive analyses have concentrated on simply determining the rigidity of moving objects as a precursor to more complex recognition schemes[13]. Others go further and attempt to fit models (2D or 3D) to objects[5, 8] in order to analyse their motions. More complex schemes attempt to interpret and characterise particular motions[4, 3] such as walking, running, sitting, doing push-ups, etc.

With reliable flow vectors for every pixel in a *blob* it becomes possible to track individual pixels from frame to frame. This capability can be employed to cluster pixels into "body parts" for model-based motion analyses (such as pfinder[14] and W^4[5]). It also means that an object's rigidity can be determined by calculating *residual flow* – that is the motion of "body parts" relative to the *blob's* gross motion. It is clear that a human or animal will have limbs moving relative to each other and to the gross body motion whereas a vehicle will not. By clustering *residual flow* vectors, it is even possible to extract these "body parts". Finally, a property called *pixel contention* is introduced in this paper to analyse occlusion. When flow is computed for an object that is occluded either through self-occlusion (such as an arm swinging in front of the body) or by another body (such as a person walking behind something else), some of the pixels will "disappear" from one frame to the next causing competition between pixels or invalid flow vectors. This *pixel contention* can be measured and used to determine when such occlusions are occurring.

Section 2 of this paper describes *blob* detection and tracking. Section 3 describes the new *dynamic region matching* algorithm for computing a dense flow field within the pixels of a moving entity. Section 4 describes how *residual flow* can be used to determine the rigidity of a moving object. Section 5 describes the *pixel contention* measure of occlusion and explains how it can be applied to determine self-occlusion within an object or mutual occlusion between two objects.

2 Detection and Tracking of Moving *Blobs*

Figure 1. *Moving* blobs *are extracted from a video image.*

Detection of *blobs* in a video stream is performed by the method described in [4]. This is basically a process of background subtraction using a dynamically updating background model.

Firstly, each frame is smoothed with a 3×3 Gaussian filter to remove video noise. The background model $B_n(x)$ is initialised by setting $B_0 = I_0$. After this, for each frame, a binary motion mask image $M_n(x)$ is generated containing all moving pixels

$$M_n(x) = \begin{cases} 1, & |I_n(x) - B_{n-1}(x)| > T \\ 0, & |I_n(x) - B_{n-1}(x)| \leq T \end{cases} \tag{1}$$

Where T is an appropriate threshold. After this, non-moving pixels are updated using an IIR filter to reflect changes in the scene (such as illumination)

$$B_n(x) = \begin{cases} B_{n-1}(x), & M_n(x) = 1 \\ \alpha I_n(x) + (1-\alpha)B_{n-1}(x), & M_n(x) = 0 \end{cases} \tag{2}$$

where α is the filter's time constant parameter. "Moving" pixels are aggregated using a connected component approach so that individual *blobs* can be extracted. An example of these extracted *blobs* is shown in figure 1.

Tracking a *blob* from frame to frame is done by the template matching method described in [9]. The important feature of this tracker is that, unlike traditional template tracking which has a tendency to "drift" when the background becomes highly textured, this method only matches pixels which have been identified as "moving" by the motion detection stage - thus making it more robust.

2.1 Tracking Through Occlusion

When two objects being tracked are predicted to occlude

$$(x_{B0} + \vec{v}_{B0}\delta t) \approx (x_{B1} + \vec{v}_{B1}\delta t) \tag{3}$$

(where x_{B0}, $\vec{v}_{B0}$, x_{B1} and $\vec{v}_{B1}$ are the positions and velocities of the two objects respectively, and δt is the time between consecutive frames) it becomes hard to track them using the template matching algorithm of [9], not because the template matching fails, but because it becomes difficult to update a template as it will be corrupted by pixels from the other object. In fact, when two *blobs* are close to occluding, the motion detection algorithm will detect them as only a single *blob*!

The problem can be addressed by storing the previous views of each object over time. When an occlusion is about to occur, synthetic templates can be generated to use in the template matching process. If the object is determined to be rigid by the method of section 4, then the latest view is used as a synthetic template while the occlusion is occurring. On the other hand, if the object is determined to be non-rigid, then its periodicity is determined by a process akin to Selinger and Wixson[13]. The latest view is matched with the previous views to find the best synthetic template. Then, as the occlusion continues, new synthetic templates are generated by stepping through the list of previous views.

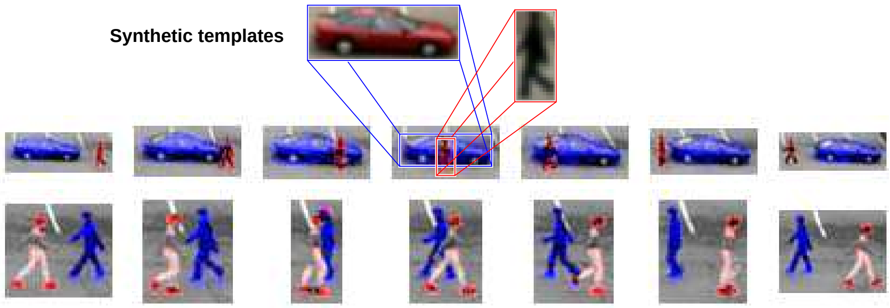


Figure 2. *Two examples of tracking through occlusion. The* blobs *are tracked using synthetic templates derived from previous views*

An example of this procedure is shown in figure 3. While the occlusion lasts, *blobs* from previous views of the object are substituted for the potentially corrupted data. These synthetic *blobs* are used for matching until the occlusion is over.

Figure 2 shows two examples of tracking through occlusion. In both cases synthetic templates are derived from previous views of both *blobs* and thus they can be disambiguated even when they are in the process of occluding. Pixels tinted red belong to the occluding object, and pixels tinted blue belong to the occluded object.

3 Optic Flow by *Dynamic Region Matching*

To analyse the motion of a character in a video stream it is necessary to acquire a dense, accurate flow field over that object, perhaps even a flow vector at every pixel. To obtain a legitimate flow vector for every pixel in an area, region-based matching is the obvious choice. But to make sure the flow vectors are realistic, it is necessary to have enough texture in the region to ensure a good match. This can be achieved by using a dynamic region size similar to the approach of Okutomi and Kanade[12]. The idea is to use edge gradients as a measure of *information* and, at every pixel position, grow the support region until there is enough edge gradient *information* to justify matching. Furthermore, flow needs to be computed only for pixels contained within the moving object. Consequently, this particular implementation is only valid for video streams derived from stationary cameras, or streams which have been stabilised.

3.1 The Region Matching Algorithm

Consider a stabilised video stream or a stationary video camera viewing a scene. The returned image stream is denoted $I_n(x)$ where I is a pixel intensity value, n is the frame number and x represents a pixel position in the image (i, j). If images in the stream are δt (time) apart, then a particular pixel in the image will move by a distance $\vec{v}(x)\delta t$ where $\vec{v}(x) = (v_i(x), v_j(x))$ is the 2D image velocity of that pixel. This can be found by matching a region in I_n to it's equivalent region in I_{n+1} by minimising a distance function $D(x; d)$ where d represents a linear translation (d_i, d_j).

Firstly, a region $W(x)I_n(x)$ is defined by multiplying I_n with a 2D windowing function $W(x)$ of size (W_i, W_j) centered on pixel x. This region is then convolved with the next image in the stream I_{n+1} to produce a correlation surface $D(x; d)$. The range of values over which the convolution is performed $d \in [d_0, d_1]$ must be specified.

$$D(x; d) = \sum_{i=1}^{i=W_i} \sum_{j=1}^{j=W_j} \frac{|W(i,j)I_n(i,j) - I_{n+1}((i,j)+d)|}{||W(x)||} \tag{4}$$

where $||W(x)||$ is a normalisation constant given by

$$||W(x)|| = \sum_{i=1}^{i=W_i} \sum_{j=1}^{j=W_j} W(i,j) \tag{5}$$

Figure 4 shows a typical correlation surface for a region match. The minimum of $D(x; d)$ is then computed to sub-pixel accuracy by approximating the surface with a quadratic function $\hat{D}(x; d)$.

The true pixel displacement $d_{\min}$ is taken as the minimum of the approximate surface

$$d_{\min} = \min_d \hat{D}(x; d) \quad (= \vec{v}(x)\delta t) \tag{6}$$

3.2 Computing Local Flow

Having established a track between a *blob* Ω_n in I_n and a *blob* Ω_{n+1} in I_{n+1} and the gross velocity $\vec{v}_B$ (from the tracker) between them, it is possible to determine the flow of every "moving" pixel. The idea is to take a small region around each pixel in Ω_n and match it with its equivalent

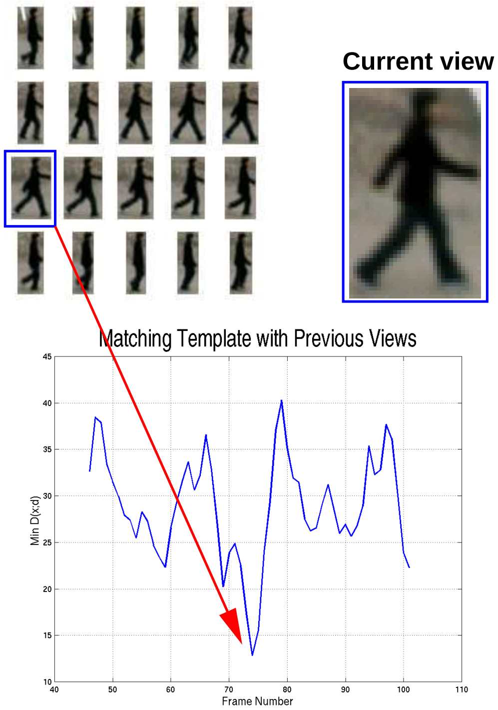


Figure 3. *Template matching the current view of a* blob *with previous views allows a synthetic template to be used when it is occluded.*

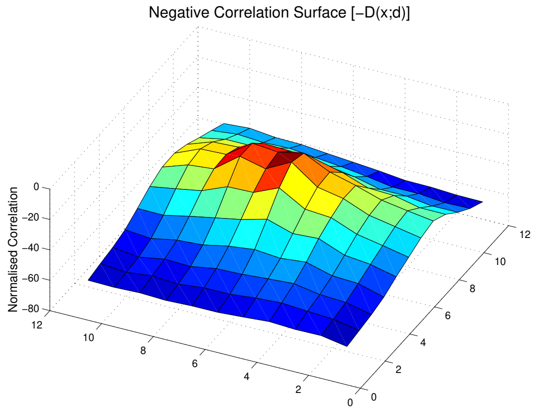


Figure 4. *A typical correlation surface (inverted for easier viewing)*

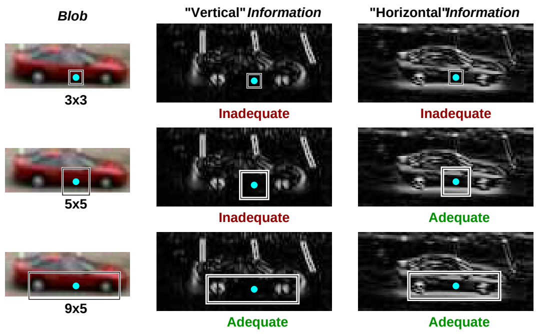


Figure 5. *Growing the region around a pixel to ensure adequate information content in both vertical and horizontal directions.*

region in Ω_{n+1}. The method used is similar to Anandan's[1] method with several distinctions. Anandan uses a pyramid to perform matching. In this implementation, a pyramid is unnecessary as the number of pixels for which flow is computed is not large. In Anandan's work, the Laplacian of the image is used. This provides a confidence measure for each flow vector. If there is not enough information in a region, the confidence on the flow vector is low. In this implementation, the image itself is used and confidence is measured using the content of the region. Anandan uses fixed 3×3 regions for matching. Here, as in Okutomi and Kanade[11], the regions are dynamic and are grown until there is enough *information* to ensure a reliable match.

The flow computation per pixel is a two-pass process. Firstly, an appropriate support region is found to ensure that enough *information* is present to obtain a valid flow vector. And then a region matching is performed to compute the flow vector.

3.2.1 Computing the Support Region

To ensure a good match between regions, it is essential that enough *information* is available in both horizontal and vertical directions around the central pixel x. So a support region around x is iteratively grown and the enclosed *information* content is measured until it reaches a predetermined threshold T_I. To measure horizontal and vertical *information*, a Sobel filter is used. Prior to the matching process, two images $S_H(x)$ and $S_V(x)$ are computed from I_n by filtering using the standard Sobel operators.

$$\begin{aligned} S_H(x) &= \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} * I_n(x) \\ S_V(x) &= \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * I_n(x) \end{aligned} \tag{7}$$

Starting with a 3×3 support window $W(x)$ centered on x two *information* measures are calculated: vertical image *information* E_V; and horizontal image *information* E_H

$$E_H = \sum_i \sum_j W(i,j) S_H(i,j) \tag{8}$$

$$E_V = \sum_i \sum_j W(i,j) S_V(i,j) \quad (9)$$

The algorithm for determining the ultimate window size is as follows

Do
Calculate E_H by equation 8
Calculate E_V by equation 9
If $E_H < T_I$
Increase W_i by 2 pixels
If $E_V < T_I$
Increase W_j by 2 pixels
Until both E_H and E_V are $> T_I$

Figure 5 shows a graphical representation of how the algorithm is applied to select an appropriate support region around a pixel.

3.2.2 Computing the Flow Vector

Motion of individual pixels can be modeled as as the gross motion of the *blob* $\vec{v}_B$ plus some residual pixel motion $\vec{v}_R(x)$. Thus

$$\vec{v}(x) = \vec{v}_B + \vec{v}_R(x) \quad (10)$$

If it is assumed that the residual motion of pixels within Ω_n is not very great, then the correlation surface $D(x; d)$ to compute the flow vector for a given pixel x need only be evaluated over a small range around $d = \vec{v}_B \delta t$.

Using the 2D windowing function $W(x)$ as determined by the algorithm of section 3.2.1, the flow $\vec{v}(x)$ for pixel x can be computed using the method of section 3.1. One limitation of this method is that background texture can corrupt the flow vector calculation. To ameliorate this, the elements of $W(x)$ can be weighted to give preference to "moving" pixels (expressed as a binary image M_n), for example

$$W'(x) = W(x)(1 + M_n(x)) \quad (11)$$

Anandan[1] uses the curvature of the correlation surface around the minimum as a measure of confidence in the flow vector. This is a reasonable choice if the *information* content of the region is unknown. Homogeneous regions will be characterised by correlation surfaces with high radii of curvature (infinite in the ultimate case). However, in this implementation, the *information* content is known, so a simpler metric for confidence is used. Here, the value of the correlation surface's minimum $D_{\min}(x; d)$ is taken as a confidence measure. Regions which match well will have low values of $D_{\min}(x; d)$ whereas regions which do not match well will have higher values. This property can be used in outlier rejection and to calculate *pixel contention* as in section 5. Figure 6 shows the flow fields and confidence measures for two different *blobs*.

4 Rigidity Analysis by *Residual Flow*

Given the gross motion of the moving body $\vec{v}_B$ as calculated in section 2, and the flow field $\vec{v}(x)$ for all of the pixels in that body, it is possible to determine the velocity of the pixels relative to the body's motion $\vec{v}_R(x)$ by simply subtracting off the gross motion

$$\vec{v}_R(x) = \vec{v}(x) - \vec{v}_B \quad (12)$$

to find the *residual flow*.

It is expected that rigid objects would present little *residual flow* whereas a non-rigid body such as a human being would present more independent motion. When the average absolute *residual flow* per pixel

$$\overline{v}_R = \frac{\sum_{i=1}^{X} \vec{v}_R(x)}{X} \quad (13)$$

(where X is the number of pixels in the *blob*) is calculated, it not only provides a clue to the rigidity of the object's motion, but also its periodicity. Rigid objects such as vehicles display extremely low values of $\overline{v}_R$ whereas moving objects such as humans display significantly more *residual flow* that even displays a periodic component.

Figure 7(a) shows the *residual flow* from two objects. Clearly, there is a large amount of independent pixel motion in the case of the human, and there is almost none in the case of the vehicle. A simple clustering based on histogramming the *residual flow* vectors clearly shows groupings of these flow vectors and could facilitate the extraction of "body parts" such as arms and legs. Figure 7(b) shows how *residual flow* can be used a measure of rigidity for humans and vehicles.

5 Occlusion Analysis by *Pixel Contention*

Many researchers have noted problems with tracking objects or their component parts through occlusions. One of the big selling points of the *Condensation* algorithm of Isard and Blake[6] is that it successfully tracks objects' shapes through deformations and occlusions. The patterns of optic flow can also be used as a key to object occlusions, whether they be caused by self-occlusion, or occlusion with other bodies. A *pixel contention* metric can be used to detect when occlusion is occuring, and even extract the spatial ordering. The object in front will display less *pixel contention* than the object in the rear.

When occlusions happen in the 3D world, they appear as 2D deformations. 2D deformation also occurs if the object changes size or "looms". In this paper, only occlusion is considered. When this occurs, the total number of pixels on the object decreases and there is contention between pixels

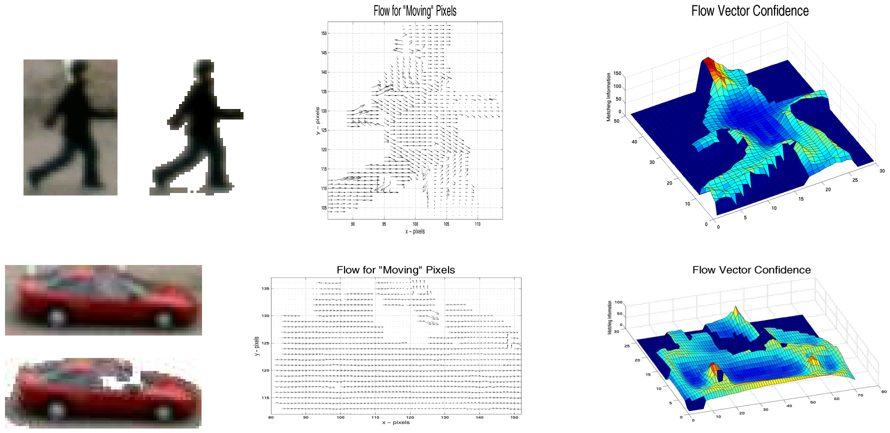


Figure 6. *The flow fields and confidence values for two different* blobs. *Notice that confidence is higher on the edges of the figures where* information *content is greater.*

for good matches as shown in figure 8. In some cases, multiple pixels in I_n match with single pixels in I_{n+1}, and in others, pixels in I_n do not have good matches in I_{n+1}. This *pixel contention* property P_c provides a good measure of occlusion. *Pixel contention* can be measured by counting the number of pixels in Ω_n which either have flow vectors terminating at a common pixel of Ω_{n+1}, or are poorly matched to pixels of Ω_{n+1}. This value can be normalised by dividing by the size X of Ω. A contended pixel x_c exists if

$$\exists(x_0, x_1) \ : \ \begin{cases} x_0 + \vec{v}(x_0)\delta t = x_c \\ x_1 + \vec{v}(x_1)\delta t = x_c \end{cases} \tag{14}$$

or

$$\min D(x_c; d) > T_c \tag{15}$$

where T_c is a threshold for determining a valid region match. And

$$P_c = \frac{\#\{x_c\}}{X} \tag{16}$$

When the first condition occurs (equation 14) the flow vector is chosen which minimises the $D(x; d)$s. That is, the vector $x \to x_c$ is chosen such that

$$x = \min_x [D(x_0; d), D(x_1; d)] \tag{17}$$

When the second condition occurs (equation 15) the flow vector for that pixel is simply ignored.

When applying *pixel contention* to a single target, it is observed that rigid bodies, such as vehicles, do not exhibit as much as non-rigid bodies, such as humans. Also, it is observed that if P_c is measured over time, it peaks when significant self-occlusions are occurring as shown in figure 9.

5.1 Occlusion by a Second Object

If two objects Ω and ω are occluding, it is important to know their spatial ordering. As with self-occlusion, it is expected that the occluded object will have a greater *pixel contention* than the foreground one. However, the absolute number of contended pixels may not be a good measure if one object is very large compared to the other, so pixel contention should be normalised with respect to the expected *pixel contention* for that object.

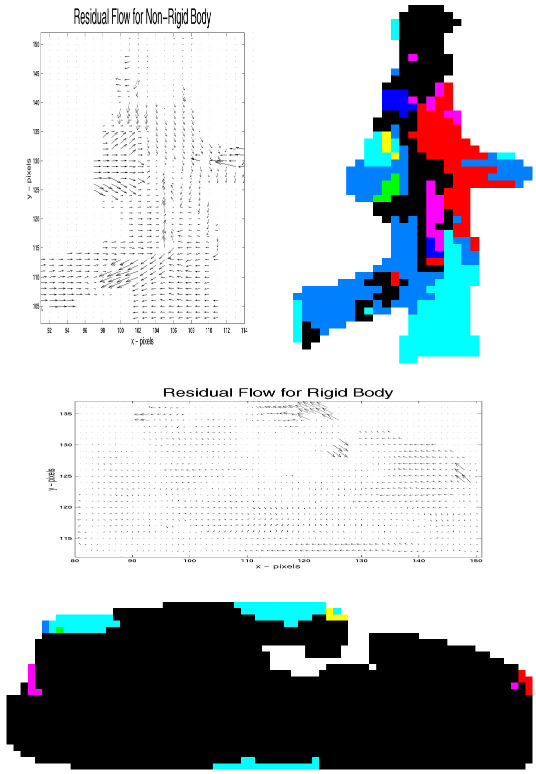
While the occlusion is occurring, the occluding *pixel contention* P_c is calculated for each of Ω and ω and normalised with respect to the average *pixel contention* for that *blob* measured prior to the occlusion. If $\overline{P_{c\Omega}}$ and $\overline{P_{c\omega}}$ are the expected *pixel contentions* of Ω and ω respectively, then a normalised occluding *pixel contention* P_o can be determined for each

$$P_o = \frac{P_c}{\overline{P_o}} \tag{18}$$

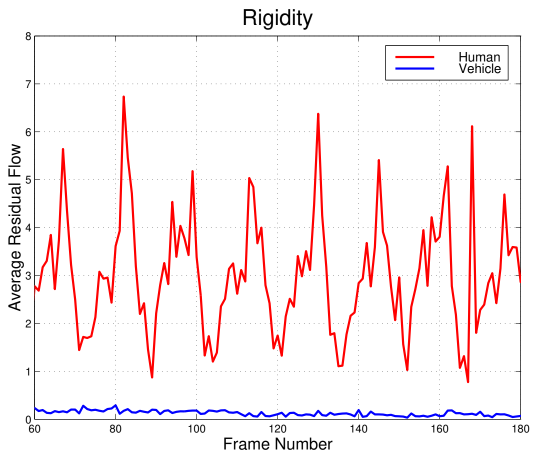
The *blob* with the larger value of normalised occluding *pixel contention* is taken as the occluded *blob* and the lower value of P_o is taken as the occluder. Figure 10 shows the normalised *pixel contention* measures for the two examples of figure 2. It is clear in both cases, that the background object displays a larger *pixel contention* than the foreground object as expected.

6 Discussion and Conclusions

Optic flow has been a research topic of interest for many years. It has, until recently, been largely inapplicable to real-time video applications due to its computationally expensive nature. This paper has shown how a new, reliable



(a)



(b)

Figure 7. *(a) The* residual flow *computed for the two* blobs *of figure 6. Also, a primitive clustering is shown. Clearly, arms and legs are apparent in the human clustering whereas only one significant cluster is visible for the vehicle. (b) The rigidity measure* $\overline{v}_R$ *calculated over time. Clearly the human has a higher average* residual flow *and is thus less rigid. It also displays the periodic nature that would be expected for a human moving with a constant gait.*

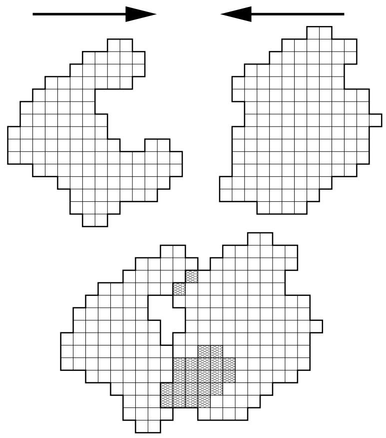


Figure 8. *When two objects occlude, pixels become contended (shown hashed). Measuring the number of such pixels can be used to measure occlusion.*

flow technique called *dynamic region matching* can be combined with a motion detection algorithm (from stationary or stabilised camera streams) to allow flow-based analyses of moving entities in real-time. If flow vectors need only be calculated for "moving" pixels, then the computation time is greatly reduced, making it applicable to real-time implementation on modest computational platforms (such as standard Pentium II based PCs). However, the issue of detecting moving entities from mobile camera video streams remains a challenge. The procedure is outlined in figure 11

Using an Anandan[1] type of region matching algorithm with Okutomi and Kanade[11] dynamic support regions allows a very accurate computation of visual flow from even very homogeneous regions on the moving object's surface.

Once optic flow has been calculated, it can be used to provide some motion analysis primitives. The *residual flow* (see section 4) can be used as a measure of the object's rigidity. These flow vectors can even be used to grossly segment the object for further region-based or model-based processing. Also, using the flow vectors to determine *pixel contention* (see section 5) provides a measure of an object's self-occlusion, and even aids in tracking objects while they are mutually occluding each other by providing a measure of spatial ordering.

The template matching process described requires $O(N^2)$ operations per region, where N is the number of *moving* pixels. This part of the procedure is clearly the computational bottleneck. If a different tracking scheme was used, it could conceivably greatly reduce the method's overall computational load. However, if the number of *moving* pixels in a rectangular region is approximately half of the total number of pixels (which is reasonable) then using this approach reduces the computational complexity by a factor

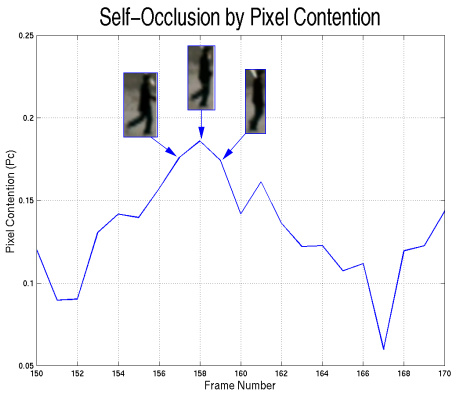


Figure 9. Pixel Contention *to measure self-occlusion. Here, the value of P_c rises considerably as parts of the human, such as arms and legs, start to occlude.*

of 4 over traditional template matching schemes. The complexity of the optical flow technique is linear in the number of moving pixels so does not become excessively unwieldy for large objects.

The presented procedure was employed on 12 video sequences from thermal sensors and 15 video sequences from daylight sensors. The sequences totalled 721 seconds and contained 47 independently moving objects. 7 false alarms were detected in the daylight imagery. These consisted of 1 case in which a tree was blown by wind, and 6 cases in which reflections of moving objects in windows cause false alarms. No false alarms were detected in the thermal imagery. Track continuity was maintained in all but 19 image frames. In each of these cases, the tracker re-acquired the target on the subsequent frame. The sequences contained 19 occlusions, all of which were correctly identified and tracking maintained through the occlusion. Admittedly, these sequences were taken under controlled conditions — it is still required to attempt this method on a wider variety of realistic surveillance imagery.

On an SGI O2 with a R10K processor, the entire process of detection, template matching, and flow computation ran at $\geq$4Hz on 320x240 monochrome images containing no more than 2 targets of $<$400 *moving* pixels. It is expected that optimising for MMX on faster commercial PC's, a greater performance could be achieved.

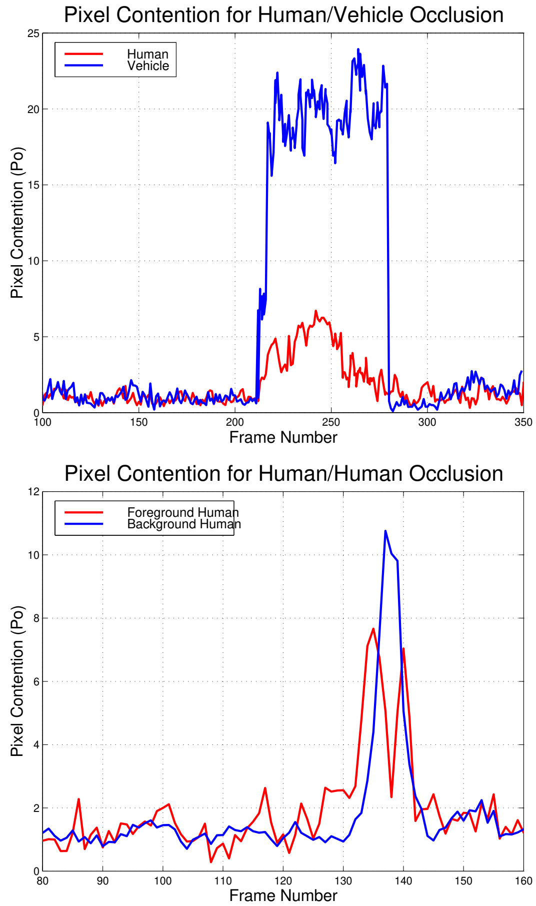


Figure 10. Pixel Contention *for the two occlusion examples of figure 2. As the vehicle is occluded, the normalised number of contended pixels increases well beyond the human. When the two humans are occluding, the occluded human displays a greater number of contended pixels. Note that at the moment of maximum occlusion, the* pixel contention *of the occluding human drops to a local minimum as it almost totally obscures the background human.*

References

[1] P. Anandan. A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision*, 2:283–310, 1989.

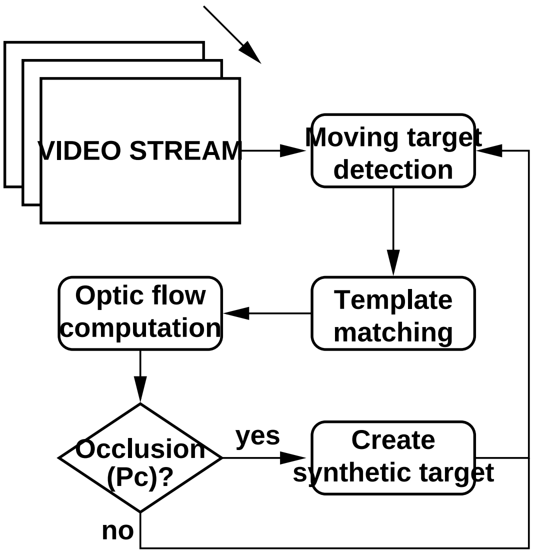


Figure 11. *The entire tracking procedure.*

[2] J. Barron, D. Fleet, and S. Beauchemin. Performance of optical flow techniques. *International Journal of Computer Vision*, 12(1):42–77, 1994.

[3] J. Davis and A. Bobick. The representation and recognition of human movement using temporal templates. In *Proceedings of IEEE CVPR 97*, pages 928 – 934, 1997.

[4] H. Fujiyoshi and A. Lipton. Real-time human motion analysis by image skeletonization. In *Proceedings of IEEE WACV98*, pages 15–21, 1998.

[5] I. Haritaoglu, L. S. Davis, and D. Harwood. w^4 who? when? where? what? a real time system for detecing and tracking people. In *FGR98 (submitted)*, 1998.

[6] M. Isard and A. Blake. Contour tracking by stochastic propagation of conditional density. In *Proceedings of European Conference on Computer Vision 96*, pages 343–356, 1996.

[7] T. Kanade, R. Collins, A. Lipton, P. Burt, and L. Wixson. Advances in cooperative multisensor video surveillance. In *Proceedings of DARPA Image Understanding Workshop*, volume 1, pages 3–24, November 1998.

[8] D. Koller, K. Daniilidis, and H. Nagel. Model-based object tracking in monocular image sequences of road traffic scenes. *International Journal of Computer Vision*, 10(3):257–281, 1993.

[9] A. Lipton, H. Fujiyoshi, and R. S. Patil. Moving target detection and classification from real-time video. In *Proceedings of IEEE WACV98*, pages 8–14, 1998.

[10] B. Lucas and T. Kanade. An interative image registration technique with an application to stereo vision. In *Proceedings of DARPA Image Understanding Workshop*, pages 121–130, 1981.

[11] M. Okutomi and T. Kanade. A multiple-baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(4), 1993.

[12] M. Okutomi and T. Kanade. A locally adaptive window for signal matching. *International Journal of Computer Vision*, 7(2):143–162, 1994.

[13] A. Selinger and L. Wixson. Classifying moving objects as rigid or non-rigid without correspondences. In *Proceedings of DARPA Image Understanding Workshop*, volume 1, pages 341–358, November 1998.

[14] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, 1997.