An Adaptive Optical Flow Technique for Person Tracking Systems

Simon Denman Vinod Chandran Sridha Sridharan

Image and Video Research Laboratory, Queensland University of Technology, GPO Box 2434, Brisbane 4001, Australia

Abstract

Optical flow can be used to segment a moving object from its background provided the velocity of the object is distinguishable from that of the background, and has expected characteristics. Existing optical flow techniques often detect flow (and thus the object) in the background. To overcome this, we propose a new optical flow technique, which only determines optical flow in regions of motion. We also propose a method by which output from a tracking system can be fed back into the motion segmenter/optical flow system to reinforce the detected motion, or aid in predicting the optical flow.

This technique has been developed for use in person tracking systems, and our testing shows that for this application it is more effective than other commonly used optical flow techniques. When tested within a tracking system, it works with an average position error of less than six and a half pixels, outperforming the current CAVIAR [1] benchmark system.

1 Overview

Tracking and surveillance applications require the segmentation of objects (regions of interest separate to the background) from the scene. Typically, systems divide the scene into two regions, foreground and background, where only the foreground contains events of interest, and the background is relatively unchanging over time. To achieve this separation of foreground and background, techniques such as motion detection or optical flow are used.

[1] The CAVIAR database, and the associated ground truth data is available for download at http://homepages.inf.ed.ac.uk/rbf/CAVIAR/

Email addresses: s.denman@qut.edu.au (Simon Denman), v.chandran@qut.edu.au (Vinod Chandran), s.sridharan@qut.edu.au (Sridha Sridharan).

Motion detection can be used to locate moving objects and segment them from the background, resulting in a binary image showing the regions of motion. Optical flow provides a means to determine and represent motion within an image sequence, where motion is represented as vectors beginning and/or terminating at pixels. Typical optical flow techniques analyse the image as a whole, calculating the flow for every pixel, regardless of motion. This results in a large amount of unnecessary processing, and can result in flow being detected in the background. While this can be overcome by separately applying motion detection, running two processes across each frame can be computationally prohibitive.

To address this, we propose a new technique that integrates optical flow directly into an adaptive background segmentation process. This ensures that optical flow is only calculated for pixels that are in motion, reducing CPU load and the presence of erroneous flow vectors. We also provide a feedback mechanism, allowing results to be reinforced, or additional information to be added based on output from a tracking system. Using this technique within a tracking system allows both motion detection and optical flow to be used, allowing more flexibility in the tracking, resulting in higher accuracy and an improved ability to handle complex situations.

Results show that our algorithm outperforms other widely used optical flow techniques for a surveillance application; where we are attempting to locate a moving person within the flow images, given expected horizontal and vertical movement. Person tracking results show that our system is able to track people accurately, with an position error of less than four pixels (median position), and that our system outperforms the current CAVIAR benchmark system.

2 Existing Work

Person detection systems such as [1][2][3] use motion detection as a first step in tracking. Once objects have been located, a variety of methods can be used to maintain tracking of an object, such as predicting the next position of the object [1][2], or using the objects colour [3], with histogram matching or colour clustering techniques.

Another common approach is to use optical flow as basis. Yamane et al.[4] proposed a method using optical flow and uniform brightness regions (a section where the optical flow cannot be detected) to track people. Okada et al.[5] uses optical flow and depth information for tracking.

These systems [4][5] rely on averaging the flow for the located object and searching for a region of similar flow vectors in the next frame. Systems use

a mix of gradient based methods [6][7] and block matching based methods [8][9]. Both methods perform best when determining flow at or around clearly defined features, and make assumptions such as constant luminance for a given region across multiple frames. As a result, when objects are not clearly defined or the lighting conditions vary, person tracking systems using these methods tend to struggle, and require additional cues to maintain tracking.

However if the only requirement of the system is to detect changes (or motion), then a background subtraction method can be used. Butler et al. [10] proposed an adaptive background segmentation algorithm where each pixel's luminance and chrominance components are modeled as a group of clusters, providing a multi-modal distribution for each pixel. The motion detection uses colour

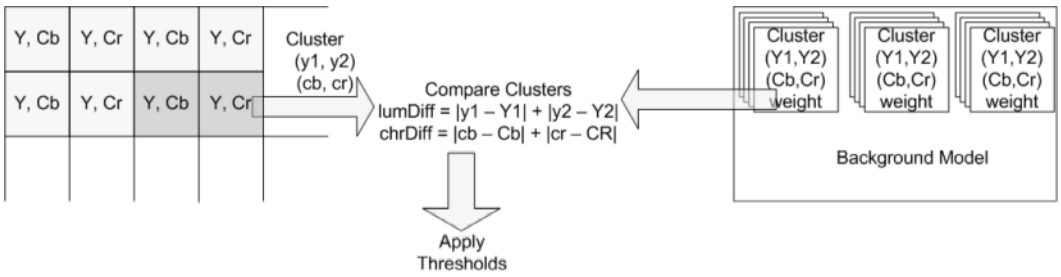


Fig. 1. Pixels are grouped into clusters, which have a centroid formed by a pair of luminance values and a pair chrominance values and a weight. Clusters built from the incoming image are compared with the appropriate group of clusters in the background model.

images in Y'CbCr 4:2:2 format as input. Pixels are matched as illustrated by figure1. Once a match is made, the matching clusters centroid and the weights of all clusters in the pixels group are adapted to incorporate the information in the matching pixel. If there is no match, then the lowest weighted cluster is replaced with a new cluster representing the incoming pixel, and the pixel is classified as being in motion.

The clusters and their weights gradually change as more frames are processed, allowing the system to adapt to changes in the background. New objects can thus be added to the scene, and over time they will be incorporated into the background model.

3 Optical Flow Algorithm

The optical flow algorithm, discussed here, is based upon the motion detection algorithm proposed by Butler et al.[10], as described in section 2.

We avoid the need for comparison with the previous frame by storing the index of the matching cluster (for each pixel) for the last frame, essentially

storing an approximation of the last frame. The accuracy of the approximation, depends on the thresholds used in the motion detection, explained in the next subsection.

Each pixel starts as being stationary. When motion is detected at a pixel, its surrounding region is examined to determine the optical flow for that pixel. The size of the area examined is governed by the maximum allowed acceleration for a pixel. For the scenes we are analysing, we have found that 7 and 3, for the x and y accelerations respectively work well (though these values are dependent on the requirements of the scene, and are likely to change for different datasets). The surrounding area is analysed outwards in rings. The centre pixel is checked first, and if a suitable match is found, searching stops. If there is no match, then the next 'ring' (at a distance of one pixel) is searched in full, and so on until a match is found. Rings may be 'truncated' to a pair of rows (or columns) if the maximum horizontal and vertical accelerations are not equal.

Flow is determined to integer precision, we do not concern ourselves with sub-pixel precision, as this level of accuracy is not required for a tracking application. Once movement for a pixel has been determined, its next position is predicted. We assume a constant velocity model, so the location of the pixel, p, in the next frame will be

$$p_x^{n+1} = p_x^n + \left(p_x^n - p_x^{n-1}\right) \tag{1}$$

$$p_y^{n+1} = p_y^n + \left(p_y^n - p_y^{n-1}\right) \tag{2}$$

where p_x^{n-1} and p_y^{n-1} are the positions of the pixel p in the previous frame; p_x^n and p_y^n are the positions of the pixel p in the current frame; and p_x^{n+1} and p_y^{n+1} are the expected positions of the pixel p in the next frame. If the pixel has previously been in movement, then the expected position is used as the position at which to start the searching.

This method of searching attempts to minimise the acceleration of a pixel, by taking the first good match when searching outwards, rather than taking the best match in the whole search area. Although we look to minimise acceleration, we do not restrict the velocity, as the pixel can continue to accelerate gradually over the course of several frames. If no suitable match for the pixel can be found, then the detection of motion at the pixel is assumed to be an error, and the motion detection is corrected.

Matching is performed between the incoming pixel (for the pixel in motion) and the last matching cluster (for the search region), simulating matching between the current and previous frame. Matching is performed using clusters, in the same way that is used for the motion detection. This process is illustrated in a numeric example in figure A.1.

An adaptive threshold is used in the motion detection to allow the system to handle different lighting conditions within the same scene. Regions that are in shadow have a different contrast to regions that are in full sun (or are lit by artificial lights) thus having different ideal thresholds for motion detection.

The adaptive threshold is dependent on the probability of the most likely cluster. The learning rate of the motion detection is such that the more frequently a colour occurs, the higher its corresponding clusters weight (representative of its probability) becomes.

$$w_k = w_k + \frac{1}{L}\left(M_k - w_k\right)[10] \tag{3}$$

where w_k is the weight of the being adjusted; L is the inverse of the traditional learning rate, α; and M_k is 1 for the matching cluster and 0 for all others. We assume that the more likely the probability, the more consistent and stable the background colour is, allowing a tighter threshold to be applied. This process is shown in a numeric example in figure A.1.

$$T = T_{max} - (w_{max} \times (T_{max} - Th_{min})) \tag{4}$$

where T is the threshold to be used for matching the pixel; Th_{max} is the maximum threshold; w_{max} is the weight of the highest weighted cluster; and Th_{min} is the minimum threshold. The learning rate is such that lower weighted clusters will increase in weight faster than higher weighted clusters, so that if the threshold is pulled too low resulting in motion being detected, the weight of the large cluster will be lowered substantially, returning the pixel to a state of no motion. This results in the thresholds for each pixel being able to reach, and approximately remain at, a natural equilibrium. We also allow the weights to be adjusted by feedback from an external source (see figure 2). Feedback can be used to reinforce motion detection in regions of interest. Matching clusters have their weight reduced to prevent the cluster from being incorporated into the background model, while non-matching clusters have their weight increased, to tighten the threshold and increase their sensitivity to motion. This can be used to prevent a slow moving or stationary foreground object being incorporated into the background model.

$$w_k = w_k \times A^{M_k} \tag{5}$$

where w_k is the weight of the cluster; A is the adaption rate and M_k is 1 if the pixel in motion, or -1 if it is not. The feedback process is illustrated in a numeric example in figure A.1.

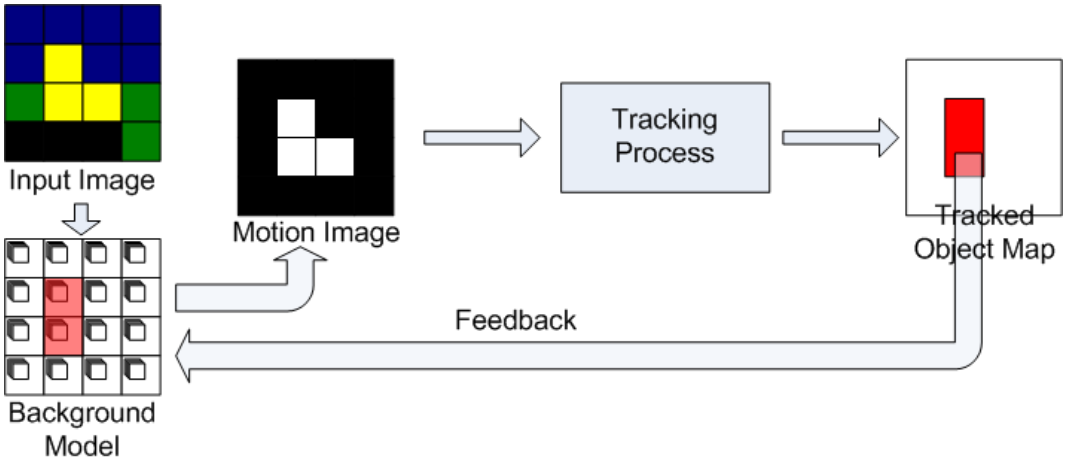


Fig. 2. Example of how the motion detection system fits into a tracking framework - Images are compared to the background model to generate a motion mask, which is used for tracking. The background model then has feedback applied to the regions containing the tracked objects.

3.2 Post Processing

We use morphological close operations to 'clean' the motion image, removing noise and filling holes in objects. However these changes do not follow through to the optical flow.

Initial Motion Image			Motion Image After Morphology			Update Process for Optical Flow Image		
M	M	M	M	M	M	Q_1	Q_2	Q_3
M		M	M	M	M	Q_4	$\frac{1}{8}\sum_1^8 Q_n$	Q_5
M	M	M	M	M	M	Q_6	Q_7	Q_8

Table 1
The Motion Image contains 8 pixels in motion (M) and one not in motion. After morphological processing, all pixels are in motion. In the optical flow image, the pixel that has changed motion state is assigned the average of the surrounding known flow values, Q_1 to Q_8. If a pixel had been changed from motion to no motion, then the flow would be set to 0.

The optical flow is updated as described by table 1. This process results in a more complete and consistent optical flow image. Whilst it may not be strictly accurate (depending on the subtle motions of the objects), it is ideal for our purposes as any holes in an object will take on a value close to that of the rest of the object, making the object detection more complete.

4 Person Tracking and Detection

The optical flow algorithm has been integrated into an existing person tracking system described in [11]. The modified system is able to use either motion detection or optical flow to detect people. To use optical flow, we need to be able to estimate the velocity of the person. As a result, the initial detection and early tracking of people is done using motion detection.

The system processes the optical flow first, detecting any people that have been tracked for sufficient time to predict velocity. Person detection using motion follows. It is preferable to detect people using optical flow, as it does not truncate a detected persons limbs by forcing them into an fixed elliptical shape, resulting in a more complete detection.

Optical flow detection is performed by using the expected horizontal and vertical movements to segment the person. Expected movement is obtained by observing the persons position and average flow over a series of frames. The use of the two modes ensures that inaccuracies in flow, or a small error in person position, does not corrupt the expected velocity. When extracting the person region, a small tolerance is allowed within the expected velocity.

$$Im_o = ((v_x - t_x) < H_f < (v_x + t_x)) + ((v_y - t_y) < V_f < (v_y + t_y)) \quad (6)$$

where H_f and V_f are the horizontal and vertical flow images; v_x and v_y are the expected movements; t_x and t_y are the allowed tolerances for the velocities and Im_o is the extracted object image. Small, isolated, regions are removed from the object map. The located object is tested for a match to the person that was being detected. We use a modified fit equation, that only considers position and shape (direction is considered via the optical flow segmentation).

$$Fit_{Object} = \frac{Fit_{Position}}{Fit_{Shape}} \quad (7)$$

Provided there is a match, the detected object is assigned to its intended person. Regions that are detected by this process are removed from the motion image, as this motion has been accounted for.

5 Results

The optical flow algorithm described here is intended for use in an intelligent surveillance system. Testing was conducted with this in mind. Four tests were carried out. The first looked at the motion detection performance; the second

Algorithm	High Contrast (GL210)		Medium Contrast (GL150)		Low Contrast (GL80)	
	False Negative	False Positive	False Negative	False Positive	False Negative	False Positive
Butler	0.22%	24.20%	0.18%	38.96%	0.20%	29.01%
Ours	0.25%	20.01%	0.24%	25.34%	0.25%	19.50%

Table 2
Motion Detection Performance

at the performance of the optical flow; the third at the speed of the system and the fourth at its performance when used in a tracking system.

The CAVIAR database was used to provide input for the testing. This database provides typical surveillance footage with ground truth information.

5.1 Motion Detection Performance

Whilst optical flow is in many ways a more informative method of motion detection, from the point of view of our surveillance application it is important that the motion detection performs well. We compare Butler's[10] original system against ours to determine what effect our changes have had on the motion detection. We used a portion the ACV Motion Detection Database [2] to evaluate our system. We used three sets at different contrasts from sequence 3 (office environment) to compare the systems. Table 2 shows that there is a performance increase when using our algorithm. There is a small increase in the false negative rate, but a significant decrease in the false positive rate as a result of the variable threshold. Thresholds for the detectors were set the same across the three tests. Butler's algorithm had thresholds set to 80 and 50 for luminance and chrominance respectively, while ours used 30 and 20 as the minimum luminance and chrominance and 150 and 120 as the maximum. Each system used 6 clusters and had L (inverse learning rate) set to 9.

5.2 Optical Flow Performance

To evaluate the performance of our optical flow algorithm, we attempt to extract people from a test sequence. Expected motion is determined using the ground truth data from the CAVIAR database. We use the difference between the median locations in the previous and current frame as the expected average velocity of the object. Extraction is performed by looking for regions with the expected horizontal movement, regions with expected vertical movement and applying a logical 'or' operation to the two images.

$$Im_{obj} = (H_{Flow} == v_x) + (V_{Flow} == v_y) \tag{8}$$

[2] This database was provided by Advanced Computer Vision GmbH - ACV

where H_{Flow} and V_{Flow} are the horizontal and vertical flow images; v_x and v_y are the expected movements and Im_{obj} is the extracted object image. We compare the performance of our algorithm with that of three other optical flow algorithms; the Lucas-Kanade [6] algorithm, the Horn-Schunck [7] algorithm, and a block matching algorithm; from the OpenCV library [3] . For these other optical flow techniques, the input images were first converted to gray scale. Due to practical considerations, we have not hand segmented the people within the images to test the segmentation performance. Instead, we have simply visually compared the performance of the various algorithms. Figure 3 shows

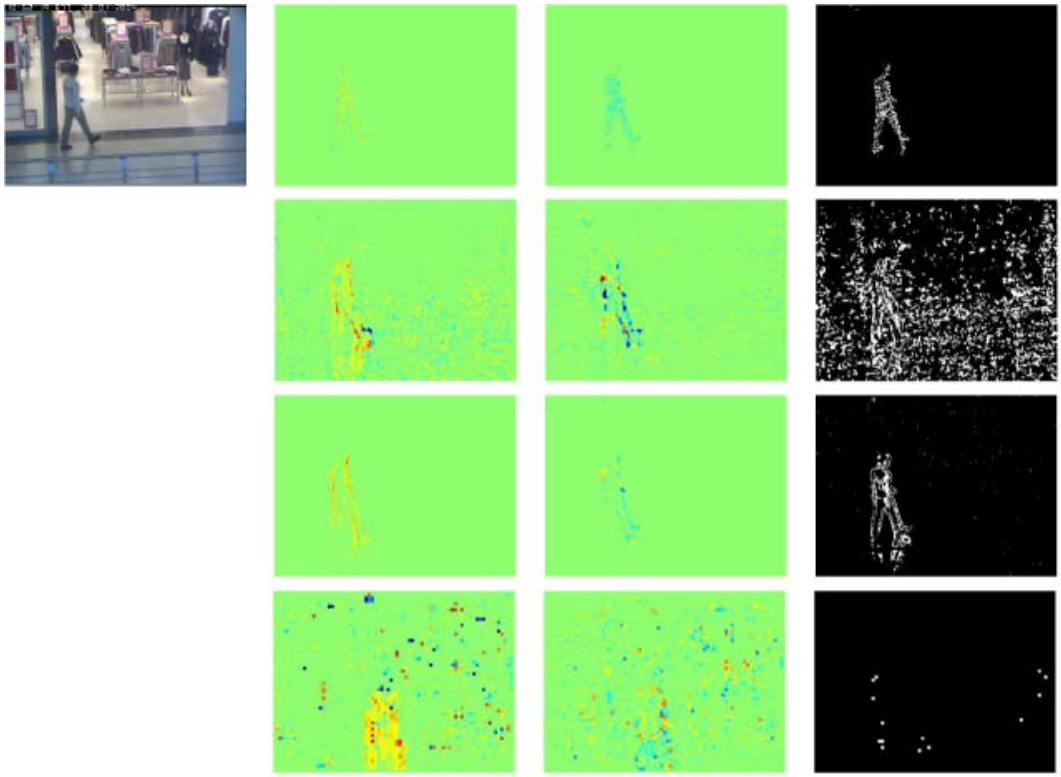


Fig. 3. Optical Flow Performance - Performance of our algorithm and three other methods. Top row shows our algorithm. Second row shows the Lucas-Kanade method [6], third row shows the Horn-Schunck method [7] and the fourth row shows a simple block matching method. Columns are (from left to right) the input frame, horizontal flow, vertical flow and extracted object image, based on the movement of the person obtained from the ground truth data supplied with the dataset. The technique was compared over a whole sequence, but results are only shown for a single frame due to size constraints

the extraction results from a single frame of our test sequence. As figure 3 shows, our algorithm is significantly better at extracting a moving object from the scene, however the segmentation is unable to extract the entire object, as

[3] The Open Source Computer Vision Library is used courtesy of the Intel Corporation and is available for public download from the World Wide Web at "'http://www.intel.com/research/mrl/research/opencv/"'.

Method	Motion Detection	Adaptive Optical Flow	Lucas-Kanade	Horn-Schunck	Block Matching
ms/Frame	16.0	20.8	33.6	30.2	192.0

Table 3
Processing Times per Frame

not all pixels within the object meet the flow criteria. This could be overcome by either using a morphological close operation, or adding a tolerance to the object extraction (i.e. detect pixels that fall within a range of flow values).

The other methods suffer from discontinuities around the edge of the person (moving object), and struggle with patches of movement that are a single colour (i.e. the persons clothes). Perhaps their biggest problem however is that they fail to distinguish background from foreground, resulting in the detection of movement in the background. These errors are brought about by the assumptions made by these techniques. Due to the lighting in the scene, there are slight fluctuations in the colour of background regions from frame to frame. Lucas et al.[6] and Horn et al.[7] use spatial intensity gradient information, where as the block matching technique uses correlation between image regions to obtain the flow. However both methods rely on the intensity of corresponding regions in the images being very similar, and the small fluctuations result in the average intensity of these corresponding regions varying from frame to frame. When this occurs on a uniform, featureless surface (i.e. floors, walls), these fluctuations can result motion being detected.

As is shown by the extracted object images, these errors result in pieces of background being detected as part of the person, which could lead to large inaccuracies in tracking. Whilst this could be overcome by masking the optical flow with a motion detector, this would only produce results that are at best as good as our algorithm, for a large increase in processor requirements.

5.3 Speed

With this algorithm aimed at use in surveillance, speed is critical. Table 3 shows the average time taken per frame for a 2000 frame video sequence for the original motion detection; our algorithm; and the Lucas-Kanade, Horn-Schunck and block matching optical flow algorithms. This test was performed on a 3Ghz workstation. Input images were 384x288 pixels in size and were loaded from disk for the tests. As the results show (see table 3), all except the block matching algorithm are capable of real time performance, however the original motion detection, and our algorithm significantly outperform the Lucas-Kanade and Horn-Schunck algorithms.

Dataset	Track	X-Error	Y-Error	Occurrences	Motion Detections	Op-Flow Detections	Predictions	False Detections	Tracking Errors
EECP1	1	1.59	1.77	152	85	44	22	0	0
	2	1.52	4.32	114	35	73	5		
OLS1	1	2.14	3.59	156	37	78	40	0	0
OLSR1	1	1.18	5.59	280	136	135	8	1	0
	2	2.17	1.80	36	12	23	0		
OSE1	1	1.99	6.95	528	222	279	26	5	0
	2	1.57	4.35	145	27	89	28		
OSE2	1	4.13	3.96	158	81	50	26	8	0
	2	1.68	10.21	725	327	386	11		
OSME2	1	1.56	6.93	896	307	504	14	4	0
OSOW1	1	1.28	9.16	366	210	118	37	21	0
	2	1.70	6.78	780	434	320	25		
	3	3.36	4.09	157	68	46	42		
	4	2.87	7.04	200	74	88	37		
Overall Performance	N/A	1.56	6.16	4693	45.6%	47.6%	6.8%	39	0

Table 4
Our System Performance - Testing was conducted on seven datasets, columns show the average x position error and y position error for the median (centre of bounding box) pixel, number of total tracked occurrences, number of detection via motion detection, number of detections via predictions, number of false detections, and the number tracking errors.

Testing has been performed using the CAVIAR database. We have used the second set of data (captured in a shopping mall) for our testing, and use the ground truth data as well as the number of incidents of incorrect detections and tracking errors to determine the quality of our results.

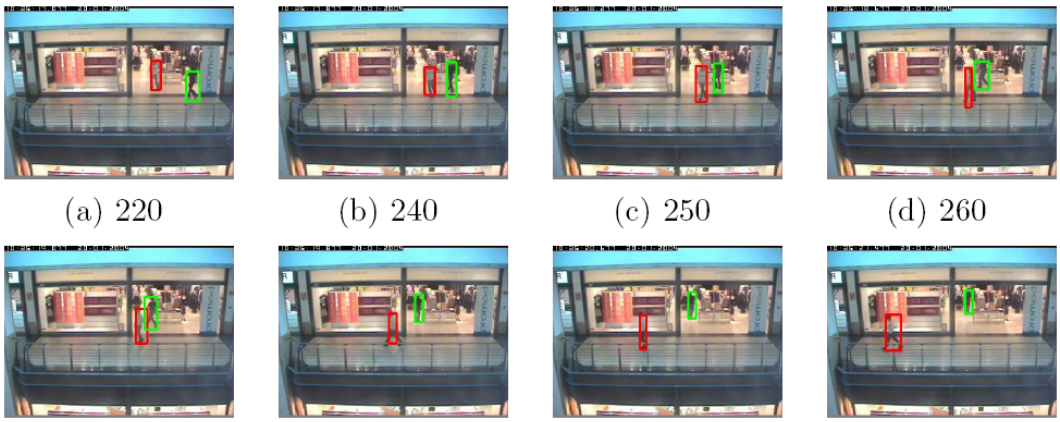
We compare our results to those provided on the CAVIAR website [4]. We use the historical performance statistics as a basis for comparison, comparing against the best result obtained to date, and most recent result. However, results are not provided for individual datasets, so these comparisons only act as a guide.

As table 4 shows, our system is able to track people with a high accuracy, averaging a euclidean error of 6.36 pixel across the seven tested datasets. The best performance achieved by CAVIAR to date has been an average error of five pixels, with the systems current performance recorded at 15. Our results also show a high detection rate, with a 6.80% missed detection rate (false negative) recorded across the five tested datasets. The CAVIAR benchmarks have a best performance of 10%, with the current system operating at over 20%.

Table 4 shows the number of tracking errors that occurred during testing.

[4] Performance data can be found at 'http://homepages.inf.ed.ac.uk/rbf/CAVIARDATA1/PERFORMANCE/'

As the results show, there were no instances of a persons track being lost or swapped, and the instance of false positives was very low, with only 39 occurrences (0.9%) in the whole of the testing, compared to a best performance of 5% for the CAVIAR system. Figures 4 and 5 show occlusion scenarios. The system is able to handle these correctly and the object ID's remain correctly assigned as the tracks cross over. Throughout the testing no errors were recorded during occlusions. Although handled correctly, the occlusions did result in an increase in the position error of the tracks (see figures 4 and 5) while the occlusion was taking place.



(a) 220 (b) 240 (c) 250 (d) 260

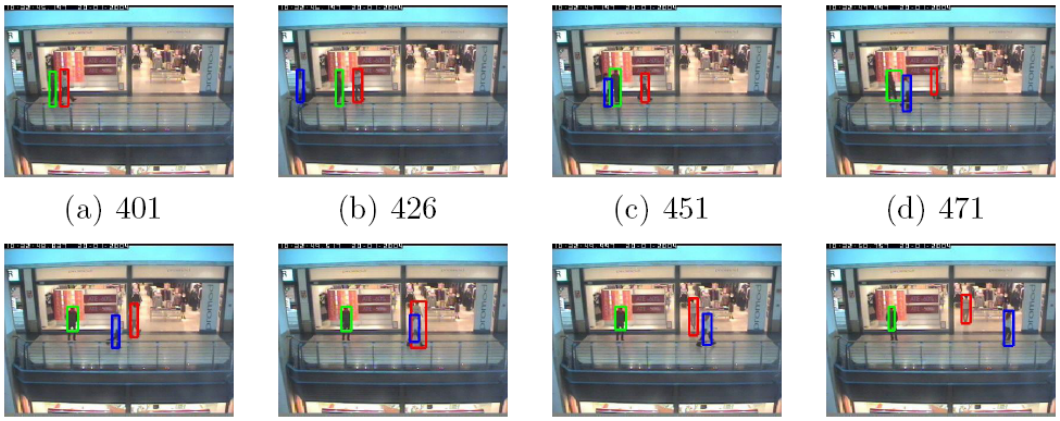
(e) 270 (f) 290 (g) 310 (h) 330

Fig. 4. System Output - Output from the dataset 'Enter Exit Crossing Paths 1'. One persons exists the shop as another enters, crossing paths. Average Euclidean error for the set is 3.26 pixels, but this increases during the occlusion (frames 250 to 290) to 3.35 pixels. Frame numbers are shown under each image.

6 Conclusions

We have described a novel optical flow algorithm designed specifically for surveillance systems where the extraction of moving objects is required. The algorithm is capable of running in real-time (25 fps) at 50% processor utilisation on a 3GHz workstation with images loaded from disk. We have shown that our algorithm sets a new benchmark for the task of moving object extraction when compared to other optical flow algorithms, and demonstrated its ability to perform well as part of a person detection system.

The algorithm is robust to small variations in intensity (such as those caused by fluorescent lighting), due to an inbuilt tolerance obtained by using a colour model that separates intensity from chrominance, and applies thresholds to each separately; and by calculating optical flow in a manner that minimises



(a) 401 (b) 426 (c) 451 (d) 471

(e) 492 (f) 509 (g) 521 (h) 540

Fig. 5. System Output - Output from the Dataset 'One Shop One Wait 1'. Two people enter the scene, one stops while one goes into the shop, a third person walks in front of the other two. Average Euclidean error for the set is 7.36 pixels, but this increases during the occlusions (frames 450-470, and 500-520) to 7.97 pixels. Frame numbers are shown under each image.

the acceleration by searching outwards from an expected position based on a constant velocity assumption.

Future work will include using the tracking system to track people and faces in surveillance, and using the optical flow to aid in super-resolution tasks to improve the quality of facial images.

References

[1] Tao Zhao and R. Nevatia, "Tracking multiple humans in complex situations", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 9, pp. 1208–1221, 2004.

[2] I. Haritaoglu, D. Harwood, and L.S. Davis, "W4: real-time surveillance of people and their activities", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 809 – 830, 2000.

[3] Wenmiao Lu and Yap-Peng Tan, "A color histogram based people tracking system", in *2001 IEEE International Symposium on Circuits and Systems*, 2001, vol. 2, pp. 137 – 140.

[4] T. Yamane, Y. Shirai, and J. Miura, "Person tracking by integrating optical flow and uniform brightness regions", in *IEEE International Conference on Robotics and Automation*, 1998, vol. 4, pp. 3267–3272 vol.4.

[5] R. Okada, Y. Shirai, and J. Miura, "Tracking a person with 3-d motion by integrating optical flow and depth", in *Fourth IEEE International Conference on Automatic Face and Gesture Recognition*, 2000, pp. 336–341.

[6] B Lucas and T Kanade, "An iterative image registration technique with an application to stereo vision", in *7th International Joint Conference on Artificial Intelligence (IJCAI)*, 1981, pp. 674–679.

[7] Berthold K.P. Horn and Brian G. Schunck, "Determining optical flow", *Artificial Intelligence*, vol. 17, pp. 185–203, 1981.

[8] J.R. Bergen, P.J. Burt, R. Hingorani, and S. Peleg, "Computing two motions from three frames", in *3rd Int. Conf. on Computer Vision*, 1990, pp. 27–32.

[9] P.J. Burt, J.R. Bergen, R. Hingorani, R.J. Kolczynski, W.A. Lee, A. Leung, J. Lubin, and J. Shvaytser, "Object tracking with a moving camera; an application of dynamic motion analysis", in *IEEE Workshop on Visual Motion*, Irvine, CA, 1989, pp. 2–12.

[10] D Butler, S Sridharan, and V. M Bove Jr, "Real-time adaptive background segmentation", in *ICASSP '03*, 2003, pp. 349–352.

[11] S. Denman, V. Chandran, and S. Sridharan, "Tracking people in 3d using position, size and shape", in *8th Internation Symposium on Signal Processing and Its Applications*, 2005, pp. 611–614.

A Worked Example

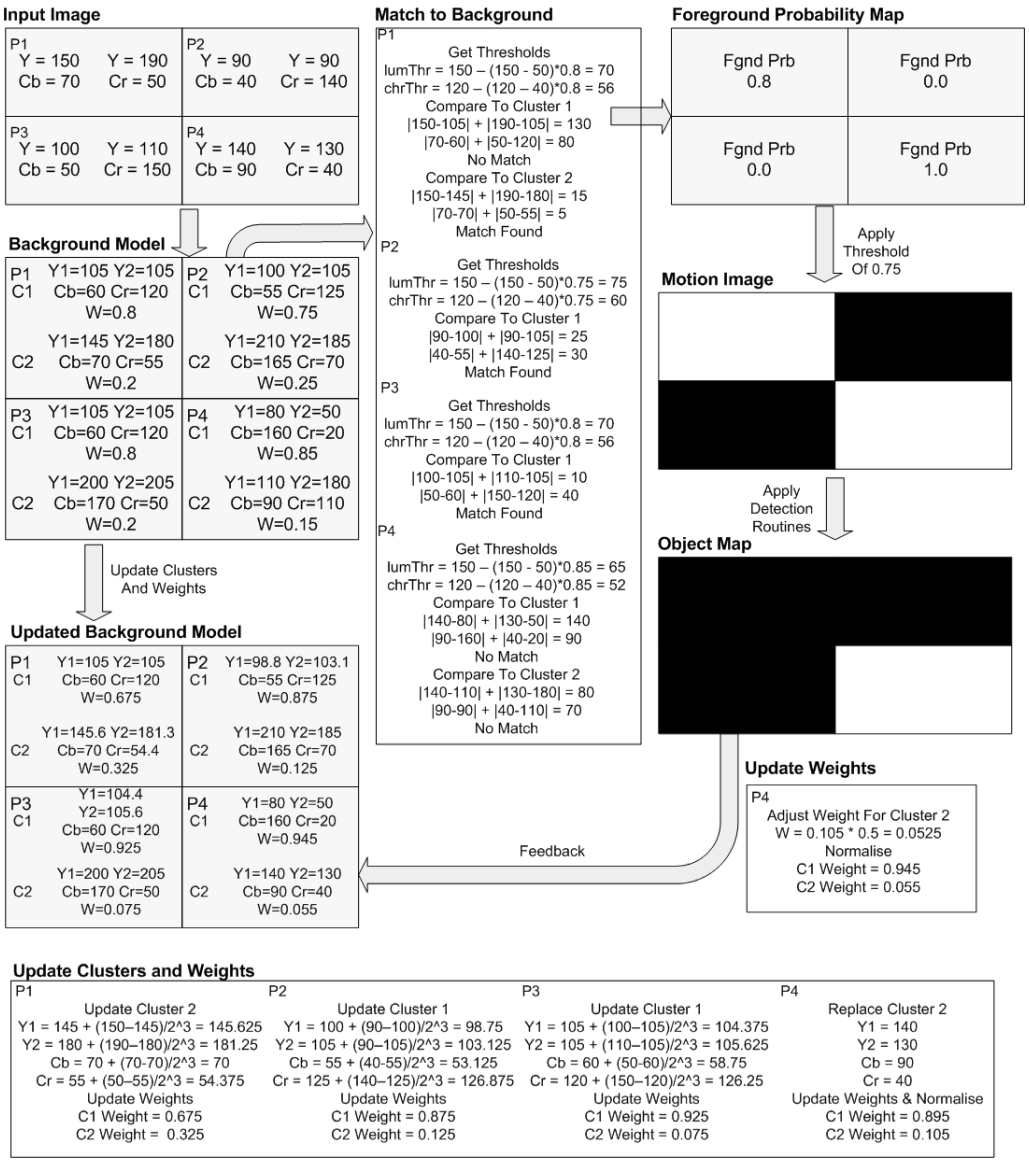


Fig. A.1. Numeric example of motion detection, showing adaptive thresholding and feedback (optical flow not shown). Example based on a system with maximum and minimum luminance thresholds of 150 and 50, maximum and minimum chrominance thresholds of 120 and 40, inverse learning rate (L) of 3, 2 clusters per pixel pair, and a foreground threshold of 0.75. It should be noted that these parameters (particularly the values of L and number of clusters) are unlikely to be used in a working system. A 4x2 pixel image is supplied as input (2x2 clusters). Two clusters are detected as being in motion ($P1$ - matched to a foreground cluster ($C2$), $P4$ - no matching cluster). Pixels $P2$ and $P3$ each match to a background cluster. Using the motion image, the object detection finds an object at a single cluster ($P4$). Feedback is used to adjust the weight of the matching cluster for $P4$.