Improved Simultaneous Computation of Motion Detection and Optical Flow for Object Tracking

Simon Denman, Clinton Fookes, Sridha Sridharan
Image and Video Laboratory
Queensland University of Technology
Brisbane, Australia
s.denman@qut.edu.au, c.fookes@qut.edu.au, s.sridharan@qut.edu.au

***Abstract*—Object tracking systems require accurate segmentation of the objects from the background for effective tracking. Motion segmentation or optical flow can be used to segment incoming images. Whilst optical flow allows multiple moving targets to be separated based on their individual velocities, optical flow techniques are prone to errors caused by changing lighting and occlusions, both common in a surveillance environment. Motion segmentation techniques are more robust to fluctuating lighting and occlusions, but don't provide information on the direction of the motion. In this paper we propose a combined motion segmentation/optical flow algorithm for use in object tracking. The proposed algorithm uses the motion segmentation results to inform the optical flow calculations and ensure that optical flow is only calculated in regions of motion, and improve the performance of the optical flow around the edge of moving objects. Optical flow is calculated at pixel resolution and tracking of flow vectors is employed to improve performance and detect discontinuities, which can indicate the location of overlaps between objects. The algorithm is evaluated by attempting to extract a moving target within the flow images, given expected horizontal and vertical movement (i.e. the algorithms intended use for object tracking). Results show that the proposed algorithm outperforms other widely used optical flow techniques for this surveillance application.**

***Keywords*-Motion Detection, Optical Flow, Object Tracking**

I. Introduction

Tracking and surveillance applications require the segmentation of objects from the scene to enable detection and tracking. This can be achieved by using techniques such as motion detection, or optical flow.

Most optical flow techniques are either gradient based methods [1], [2], or block matching based methods [3]. Gradient based methods have been preferred due to speed and performance considerations. These methods analyse the change in intensity and gradient (using partial spatial and temporal derivatives) to determine the optical flow. Block matching based methods rely on determining the correspondence between the two images, by matching 'blocks' of one image to 'blocks' of the other. Both methods perform best when determining flow at or around clearly defined features, and make assumptions of constant luminance and spatial continuity. As a result, when objects are not clearly defined (perhaps due to clutter) or the lighting conditions vary, errors can occur in the optical flow. Performance also suffers where trying to determine the flow for uniform regions where there is little or no texture. To try and overcome the limitations of these methods, Black et al. [4] proposed a robust method based around a robust estimation framework. The estimation framework reduces the outliers caused by motion discontinuities and violations of the constant luminance assumption. Other algorithms have sought to solve the problem of spatial discontinuities, either by detecting occlusions and extrapolating optical flow in occluded areas [5], by using more than two frames in flow calculations [6]. Zach et al. [7] proposed an alternative approach using the total variation regularization and the robust L^1 norm in the data fidelity term. This approach is able to preserve discontinuities in the flow field whilst offering increased robustness against illumination changes, occlusions and noise.

A problem with using optical flow for object tracking is it is difficult to initially detect the objects, particularly given that in surveillance environments optical flow algorithms often detect erroneous motion in background regions due to subtle changes in lighting or camera noise. Probabilistic tracking techniques such as particle filtering [8] or the mean shift algorithm [9] work well for tracking as they can overcome noise and erroneous flow errors, however these require an initial detection. Object tracking systems that use motion detection [10], [11] are able to more easily detect objects using the binary image that results from motion detection. However while optical flow can be used to segment overlapping objects using velocity, motion detection cannot. Ideally, motion detection could be used to perform initial object detection and optical flow used there after once the targets velocity is known, but running both algorithms is computationally prohibitive.

Denman et al. [12] proposed a combined motion segmentation/optical flow algorithm, by extending a motion segmentation algorithm [13] to simultaneously compute optical flow and motion detection. This combination ensures that optical flow is only calculated for pixels that are in motion, reducing CPU load and the presence of erroneous flow vectors. The technique was shown to result in an improvement in object tracking performance when incorporated into an

978-0-7695-3866-2/09 $26.00 © 2009 IEEE
DOI 10.1109/DICTA.2009.35

object tracking system. However the proposed technique is limited in that it can only determine flow to a two-pixel resolution in the horizontal direction, meaning flow fields are inaccurate and results are poor for slow moving objects.

We propose a new technique that integrates optical flow directly into an adaptive background segmentation process. Unlike [12], optical flow is calculated at pixel resolution. Motion segmentation information from the previous frame is used when determining flow, to ensure that no comparisons are made to background regions. Short term tracking of flow vectors is also used to improve speed and accuracy, as well as detect discontinuities. Using this technique within a tracking system allows both motion detection and optical flow to be used, allowing greater flexibility when tracking. The algorithm requires a fixed camera (required by the motion segmentation), however as the majority of surveillance cameras are fixed this is not considered to be a major limitation. The proposed algorithm is evaluated by attempting to extract a moving target within the flow images, given expected horizontal and vertical movement (i.e. the algorithms intended use for object tracking). Results show that the proposed algorithm outperforms other widely used optical flow techniques for this surveillance application.

II. Proposed Approach

The proposed algorithm uses the motion segmentation algorithm proposed by Butler et al. [13] as a basis. An overview of this algorithm is provided in Section II-A, and Section II-B discusses the proposed optical flow extension.

A. Foreground Segmentation

An efficient method of foreground segmentation that is robust and adapts to lighting and background changes was proposed by Butler [13]. This approach is similar in design to the Mixture of Gaussian's (MoG's) approach proposed by Stauffer and Grimson [14], in that each pixel is modeled by a group of weighted modes that describe the likely appearance of the pixel. Unlike the MoG's approaches, cluster structures consisting of four colour values (stored as two pairs) and a weight are used to represent the pixel modes.

The algorithm uses Y'CbCr 4:2:2 images as input, and clusters are formed by pixel pairs (see Figure 1). Each pixel in the incoming image has two values, a luminance and a single chrominance, which alternates between blue chrominance and red chrominance. Consecutive pixels are paired for such that each pair contains two luminance values, one blue chrominance value and one red chrominance value. This pairing results in motion detection being effectively performed at half the horizontal resolution of the original image.

Let $p(x_i, y_i, t)$ be a pixel in the incoming Y'CbCr 4:2:2 image, $I(x_i, y_i, t)$ where $[x_i, y_i]$ is in $[0..X-1, 0..Y-1]$ and t is in $[0, T]$. A pixel pair, $P(x, y, t)$ (where $[x, y]$ is in $[0..\frac{X}{2}-1, 0..Y-1]$) is formed from $p(x_i, y_i, t) = [y_1, cb]$

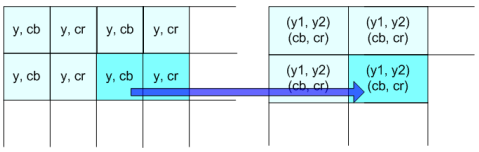


Figure 1. Motion Detection - Input Image to Clusters

and $p(x_i + 1, y_i, t) = [y_2, cr]$ to obtain four colour values, $P(x, y, t) = [y_1, cb, y_2, cr]$ (where $x_i = x \times 2$, and $y_i = y$). These four values are treated as two centroids ((y_1, y_2) and (cb, cr)). Each image pixel, $p(x_i, y_i, t)$, is only used once when forming pixel pairs $P(x, y, t)$. Pixel colour history is recorded by a set of K clusters,

$$C(x, y, t, 0..K-1) = [y_1, y_2, cb, cr, w], \tag{1}$$

which represents a multi-modal PDF. Each cluster contains four colour values (y_1, y_2, cb, cr) and a weight, w. The weight describes the likelihood of the colour described by that cluster being observed at that position in the image. Clusters are stored in order of highest to lowest weight. C_{y_1}, C_{y_2}, C_{cb}, $C_{y_{cr}}$ and C_w are defined as the four colour values and weight for the cluster C.

For each $P(x, y, t)$ the algorithm makes a decision assigning it to background or foreground by matching $P(x, y, t)$ to $C(x, y, t, k)$, where k is an index in the range 0 to $K-1$. Clusters are matched to incoming pixels by finding the highest weighted cluster which satisfies,

$$|y_1 - C_{y_1}(k)| + |y_2 - C_{y_2}(k)| < T_{Lum}, \tag{2}$$

$$|cb - C_{cb}(k)| + |cr - C_{cr}(k)| < T_{Chr}, \tag{3}$$

where y_1 and cb are the luminance and chrominance values for the image pixel $p(x \times 2, y)$, y_2 and cr are the luminance and chrominance values for the image pixel $p(x \times 2 + 1, y$, $C(k) = C(x, y, t, k)$; and T_{Lum} and T_{Chr} are fixed thresholds for evaluating matches. The centroid of the matching cluster is adjusted to reflect the current pixel colour,

$$C(x, y, t, \kappa) = C(x, y, t, \kappa) + \frac{1}{L}\left(P(x, y, t) - C(x, t, y, \kappa)\right), \tag{4}$$

where κ is the index of the matching cluster; and the weights of all clusters in the pixels group are adjusted to reflect the new state,

$$w'_k = w_k + \frac{1}{L}\left(M_k - w_k\right), \tag{5}$$

where w_k is the weight 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.

If $P(x, y, t)$ does not match any $C(x, y, t, k)$, then the lowest weighted cluster, $C(x, y, t, K-1)$, is replaced with a new cluster representing the incoming pixels. Clusters are gradually adjusted and removed as required, allowing the system to adapt to changes in the background. After the updating of weights and clusters, the cluster weights

are normalised to ensure they sum to one. Based on the accumulated pixel information, the frame can be classified into foreground,

$$fgnd = \forall(x, y, t)\ where \sum_{i=0}^{\kappa} C_w(x, y, t, i) < T_{fgnd}, \quad (6)$$

where T_{fgnd} is the foreground/background threshold and κ is the matching cluster index; and background.

B. Incorporating Optical Flow

Optical flow calculations require the previous frame to be compared to the current to determine motion. The need for comparison with the previous frame is avoided by maintaining a record of the matching cluster for each pixel for the last frame, essentially an approximation of the last frame. The accuracy of the approximation depends on the thresholds used in the motion detection.

As the algorithm is intended to function at a pixel resolution (i.e. not cluster resolution, which is down sampled by 2 in the horizontal direction), chrominance information cannot be used due to the down sampling that has taken place (YCbCr 4:2:2 input). Instead, luminance and luminance gradient are used. Gradient is calculated for each pixel and incorporated into the background model so that a cluster becomes,

$$C(x, y, t, k) = [y_1, y_2, cb, cb, y_1^{gv}, y_1^{gh}, y_2^{gv}, y_2^{gh}, w], \quad (7)$$

where y_1^{gv} is the vertical gradient for y_1, y_1^{gh} is the horizontal gradient for y_1. The clusters along the top edge of the image have a vertical gradient of 0, and those on the left edge of the image have a horizontal gradient of 0, as there is no pixel to subtract to obtain a gradient. The gradient values are updated in the same manner as the other values within the cluster, however they are not used during foreground segmentation due to the gradient of a pixel at (x, y) being affected by motion at the pixels at $(x-1, y)$ and $(x, y-1)$, which may lead to additional false positives. This is not considered a problem for the optical flow calculations as when a region is in motion, it can be assumed that the surrounding pixels are also undergoing the same, or similar, motion.

Optical flow is calculated using block matching over cluster windows at pixel resolution. Let $W(x1 : x2, y1 : y2, t)$ be the window of pixels extracted from the incoming frame centred about (x, y) (the position of the cluster the flow is being determined for), and $W(x1' : x2', y'1 : y2', t-1)$ be the window of clusters from the previous frame (to be compared to $W(x1 : x2, y1 : y2, t)$), centred about (x', y') (a possible position for the cluster in the previous frame). For a comparison to be made, the pixels in $W(x1' : x2', y1' : y2', t-1)$ must have been in motion at $t-1$. This ensures that no attempts are made to match to parts of the background. The set of clusters that are compared then becomes,

$$W_{fgnd}(x, y, t) = P(x, y, t-1) \in fgnd(t-1) \quad (8)$$
$$where\ (x, y) \in W(x1' : x2', y1' : y2', t-1),$$

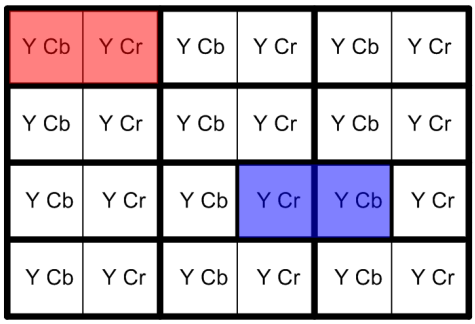


Figure 2. Matching Across Cluster Boundaries

where $W_{fgnd}(x, y, t)$ is the set of clusters for the window centred at x, y that were in the foreground last frame. The number of clusters in the set is defined as P_{count}. If this foreground condition is not enforced, then for a cluster that lies on an object boundary, part of the comparison will be performed against the background each frame. As the object is moving, the part of the background being compared to will be constantly changing, and so it can be assumed that there will be no match between these sections of the window. These poor matches may result in the whole window being ruled a poor match, impeding the ability of the system to estimate optical flow.

When performing matching to determine the flow, the area surrounding the pixel is motion 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. Each ring is searched in full, and the best match within the ring (if a match is present at all) is accepted. Rings may be 'truncated' to a pair of rows (or columns) if the maximum horizontal and vertical search distances are not equal. This search method attempts to minimise the acceleration of a pixel by taking the first good match searching outwards, rather than the best match in the whole search area. Although the approach aims to minimise acceleration (constant velocity assumption) no restriction is placed on the velocity, as the pixel can continue to accelerate gradually over the course of several frames.

Calculating optical flow at pixel level on two pixel wide blocks (the clusters) requires two different comparison routines. When matching between two compete clusters (i.e. matching at an even horizontal distance) the same matching equations used when comparing clusters for the motion detection can be reused, with gradient substituted for chrominance.

When matching across a cluster boundary, the comparison is effectively between one cluster in the current image to two in the previous image. In Figure 2, the area marked in red is the cluster in the current image we are trying to find a match for $(C(x, y, t))$, and the area marked in blue as the

previous frames cluster we are trying to match to. The blue area is actually two clusters. $C(x'-1, y', t-1)$ is defined as the left most of the two, and $C(x', y', t-1)$ is the right most. The comparison between these clusters then becomes

$$\begin{aligned} Diff_{Lum}(x,y,t) = \\ |C_{y_1}(x,y,t) - C_{y_1}(x'-1,y',t-1)| + \\ |C_{y_2}(x,y,t) - C_{y_2}(x',y',t-1)|, \end{aligned} \tag{9}$$

$$\begin{aligned} Diff_{Grad}(x,y,t) = \\ \left|C_{y_1^{gv}}(x,y,t) - C_{y_1^{gv}}(x'-1,y',t-1)\right| + \\ \left|C_{y_2^{gv}}(x,y,t) - C_{y_2^{gv}}(x',y',t-1)\right| + \\ \left|C_{y_1^{gh}}(x,y,t) - C_{y_1^{gh}}(x'-1,y',t-1)\right| + \\ \left|C_{y_2^{gh}}(x,y,t) - C_{y_2^{gh}}(x',y',t-1)\right|. \end{aligned} \tag{10}$$

When matching across a cluster boundary, $C(x'-1, y', t-1)$ and $C(x', y', t-1)$ are checked separately to determine if they were in motion at time $t-1$. If only one was, then only the portion of the comparison that involves that cluster is performed. In this instance, P_{count} is incremented by 0.5 (only half the comparison has been performed).

Matching scores for cluster windows are obtained by calculating the average error in the luminance and gradient matches (W_{Lum} and W_{Grad}) for all foreground pixels in the window,

$$\begin{aligned} W_{Lum} = \frac{1}{P_{count}} \sum_{x,y} Diff_{Lum}(x,y,t), \\ (x,y) \in W_{fgnd}(x,y,t), \end{aligned} \tag{11}$$

$$\begin{aligned} W_{Grad} = \frac{1}{P_{count}} \sum_{x,y} Diff_{Grad}(x,y,t), \\ (x,y) \in W_{fgnd}(x,y,t). \end{aligned} \tag{12}$$

W_{Lum} and W_{Grad} are compared to thresholds and if met, and if $P_{count} > 1$, then a potential match has been found (whether this is the actual match depends on what, if any, other matching windows are detected in the current search ring). If P_{count} is less that or equal to 1, there is not sufficient evidence for a match. In this case, (i.e. a match has been made to an isolated pixel), it is likely the match has been made to noise.

Once movement for a cluster has been determined, the cluster's next position is predicted. Optical flow vectors are tracked within the system using a constant velocity motion model. Optical flow information is propagated through the system from frame to frame, tracking the movement of pixels over time. For each cluster at time t that has non-zero optical flow, a prediction is propagated forward to the expected position at time $t+1$,

$$\upsilon_x(t+1) = \upsilon_x(t) + (\upsilon_x(t) - \upsilon_x(t-1)), \tag{13}$$

$$\upsilon_y(t+1) = \upsilon_y(t) + (\upsilon_y(t) - \upsilon_y(t-1)), \tag{14}$$

where $\upsilon_x(t)$ and $\upsilon_y(t)$ are the positions of the cluster p at time t. At time $t+1$, when the system is processing a cluster that has a prediction associated with it, it will use the prediction provided as a starting point for the search. Multiple predictions are allowed to be propagated forward to a single pixel. When determining flow at that pixel in the next frame, the centre point of each prediction is checked first, and the best match (if there is a match at all) is taken as the flow. If there is no match, the surrounding areas of the each prediction is analysed to find a match.

Predictions are stored with an accumulated average velocity (u_{ave} and v_{ave} for the horizontal and vertical velocities respectively), and a counter (f_{count}) to indicate how many successive frames the pixel has been observed in motion for. Average velocities are calculated using,

$$u_{ave}(t) = u_{ave}(t-1) + \frac{1}{L_{opf}}(u(t) - u_{ave}(t-1)), \tag{15}$$

$$v_{ave}(t) = v_{ave}(t-1) + \frac{1}{L_{opf}}(v(t) - v_{ave}(t-1)), \tag{16}$$

where u and v are the horizontal and vertical flows for the current frame, and L_{opf} is the learning rate for the average optical flow.

If optical flow cannot be determined for a cluster, the list of predictions for that cluster is used to estimate the flow. The prediction with the highest f_{count} is used to estimate the optical flow. A flag is set to indicate that it is a prediction, as a prediction can only be propagated through for R successive frames. R is kept small (< 3) as only a simple motion model is used. If a cluster is not detected as being in motion, no flow calculations are performed and optical flow is set to 0 for the cluster.

C. Detecting Overlapping Objects

A cluster in motion can take on one of four states:

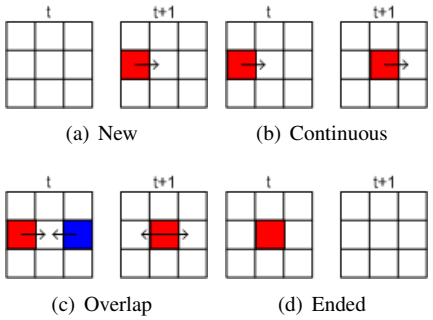
1) *New* - the first appearance of a cluster, its flow cannot be determined as it was not present in the last frame.
2) *Continuous* - the cluster is in motion and a match to the previous frame has been found.
3) *Overlap* - the cluster was in motion last frame and cannot be found this frame. The space that it should occupy this frame is occupied by another cluster.
4) *Ended* - the cluster cannot be found and there is no overlap condition.

These four states are illustrated in Figure 3.

The state, $S(x,y,t)$, of a cluster $P(x,y,t)$ is determined using the propagated optical flow information.

When motion is detected at a cluster, $P(x,y,t)$, and its optical flow ($U(x,y,t)$ and $V(x,y,t)$) is known, $P(x,y,t)$'s flow information is updated (increase counter and adjust averages), and the motion arising from the cluster in the previous frame is marked as accounted for.

At the end of the frame, any moving cluster at $t-1$ whose motion has not accounted for is either involved in an overlap,



(a) New (b) Continuous

(c) Overlap (d) Ended

Figure 3. Optical Flow Pixel States

or its motion has ended. The prediction for the pixel is checked (if there are multiple predictions, then the prediction with the highest f_{count}) and if the predicted position is occupied by motion, then an overlap has occurred. If there is no motion at the predicted position, then the motion has ended. An overlap can only occur is the cluster that is obscured has been observed for T successive frames (i.e. the motion of that cluster has been observed for several frames and is considered reliable). This helps to reduce erroneous overlaps.

The detection of overlapping pixels can aid in detecting overlaps between moving objects in a scene, particularly for objects undergoing similar motion.

III. Results

The proposed algorithm is intended for use in object tracking scenarios, where a fixed camera would be used. Its intended use is to aid in object tracking by providing an additional object detection method (the other being using the motion mask to locate suitable regions of foreground) to extract an object based on the object's expected velocity. As such, optical flow evaluation data sets such as the Middlebury data sets [1] are not suitable for evaluating this algorithm. These data sets are designed for evaluating traditional optical flow algorithms (i.e. do not require a fixed view point, do not require a set of frames to initialise a background model).

To evaluate the performance of the proposed hybrid motion detection optical flow algorithm, attempts are made to extract people from images in the CAVIAR data set [15]. As the algorithm is intended for surveillance applications and requires a fixed camera and sufficient frames to learn the background, an object tracking database is appropriate. The expected velocities of the target people are determined using the ground truth data from the CAVIAR database [15]. The difference between the median locations in the previous and current frame is used as the expected velocity of the object. Extraction is performed by finding pixels that satisfy,

$$Im_{obj} = |H_{Flow} - v_x| + |V_{Flow} - v_y| < T_{error}, \quad (17)$$

[1]The Middlebury optical flow evaluation can be found at http://vision.middlebury.edu/flow/eval/.

where H_{Flow} and V_{Flow} are the horizontal and vertical flow images; v_x and v_y are the expected movements, T_{error} is the allowed error (set to 1.5) and Im_{obj} is the extracted object image.

The performance of the proposed algorithm is compared to five other optical flow algorithms; the algorithms proposed by Denman et al [12], Lucas and Kanade [1], Horn and Schunck [2], Black et al [4] and Zach et al [7]. For [1], [2], [4], [7], the input images are converted to gray scale, and results are masked against the motion image obtained by the proposed algorithm. The target people within the images have been hand segmented to test performance. Figures 4 to 6 show the segmentation performance of the algorithms. Cropped images are shown to improve readability, image (a) in Figures 4 to 6 shows the cropped region in a red rectangle. Results in terms of false negatives and false positives are shown in Table I. Performance shown in Table I is measured over the cropped region only.

As Figures 4 to 6 and Table I shows, the proposed algorithm is significantly better at extracting a moving object from the scene. The segmentation is unable to extract the entire object, as not all pixels within the object meet the flow criteria. This could be partially overcome by applying a morphological close operation, however in cases where severe occlusions are present (i.e. Figure 6), it is likely that some of the target object will still remain undetected.

Lucas-Kanade [1] and Horn-Schunck [2] suffer from discontinuities around the edge of the person, and struggle with patches of movement that are a single colour (i.e. the persons clothes). They fail to distinguish background from foreground, resulting in the detection of movement in the background. This can be seen in the horizontal and vertical flow images, however as the object extraction using the motion image as a mask, these errors are not seen in the extracted object. Due to the fluctuating lighting in the scene, there are slight fluctuations in the colour of background regions from frame to frame. Black [4] and Denman [12] overcome this, as their approaches are more robust to small fluctuations. However both techniques fail to clearly extract the target object in Figures 5 and 6. Zach [7] performs well and is able to extract significant portions of the target objects. However, [7] detects large erroneous patches of optical flow in regions such as the shadows of the moving people (see Figure 4 (s)) and on the reflective floor (see Figure 4 (t), Figure 5 (t) and Figure 6 (t)). Whilst these do not result in any segmentation errors in the examples shown here (the regions are masked out by the motion detection and, in the case of the reflections, the flow vectors lie well outside the range of values being detected), this may not be the case in other situations.

Overlap detection is evaluated a sample sequence from the CAVIAR database [15] that contains two people overlapping. The optical flow status images that are produced are used to detect areas where there are a high proportion of flow

Image	Algorithm											
	Proposed		Denman		LK		HS		Black		Zach	
	FP	FN	FP	FN	FP	FN	FP	FN	FP	FN	FP	FN
WBS1 front, 640	1.19%	10.88%	1.27%	15.61%	0.24%	78.00%	0.31%	87.01%	0.92%	12.18%	0.86%	23.69%
OSNE2 front, 1085	0.78%	41.13%	0.35%	54.15%	0.05%	99.06%	0.01%	97.36%	0.00%	100.00%	0.19%	71.51%
OSNE2 front, 1101	1.42%	58.35%	0.36%	94.67%	0.19%	98.55%	0.07%	99.27%	0.00%	100.00%	0.42%	59.56%
Average	1.15%	19.24%	0.87%	28.07%	0.19%	82.61%	0.19%	89.49%	0.52%	31.61%	0.62%	33.11%

Table I
SEGMENTATION PERFORMANCE - FP IS THE FALSE POSITIVE RATE, FN IS THE FALSE NEGATIVE RATE.

discontinuities, likely to be caused by overlapping objects or the edge of a region of motion. Detection is performed by analysing vertical projection histograms of the different optical flow states. As such, only edges that run vertically are detected in this test.

The example sequences (see Figures 7 and 8) show the motion detection output in the top row, the flow status in the second and the original frame with the overlaps marked (shaded red bars) in the bottom row. The optical flow status images show the optical flow states (see Section II-C) as Green for *New*, Yellow for *Continuous*, Red for *Overlap*, Blue for *Ended* and Black for pixels which contain no motion.

The example sequences show that overlaps can be detected by analysis of the flow status images (see Figure 8 (j) and (k)). Several object edges are also detected (Figure 8 (l)). The detection at the edges can be attributed to increased instances of pixels in the *New* and *Stopped* states at the boundary of the person (i.e. a discontinuity). Further analysis of these detection, analysing the ratio of *Overlap* pixels to *New* and *Stopped* can separate these edge detections from overlaps.

The flow status images show an greater concentration of *Overlap* pixels being detected when an occlusion is occurring. There are also isolated overlap pixels detected elsewhere, which can be partially attributed to the data set. As can be seen in the motion detection results, the motion detection performs poorly around the legs of the people (due to the dark edge at the bottom of the shop front, which is a very similar to colour to the pants worn by all subjects) and results in large amounts of *New* motion being detected about the legs. This results in new motion being detected at the legs every frame, some of which forms false overlaps in later frames. Despite this, in the three samples shown only 2 false object edges are detected.

The speed of the proposed algorithm is benchmarked using a 1000 frame segment of the CAVIAR database [15] (352x288 size images). Benchmarking is performed on a 2.4GHz Intel Core 2 Quad-Core CPU, running algorithms on a single core. Table II shows the throughput of each of the algorithms. As can be seen, the proposed algorithm is capable of real time processing and offers significantly better performance for the target application than [1], [2], [12], [4]. Performance for [7] is not given as a matlab version of algorithm was used in this evaluation [2]. However, it should be noted that [7] is capable of real time performance when implemented on a GPU.

Algorithm	Throughput (fps)
Proposed	27.43
Denman	32.67
Lucas & Kanade	43.83
Horn & Schunck	37.17
Black	0.33

Table II
AVERAGE THROUGHPUT

IV. CONCLUSIONS AND FUTURE WORK

This paper has proposed a new algorithm for simultaneously computing motion segmentation and optical flow. The algorithm is aimed at object tracking applications, using a fixed camera. The proposed algorithm is able to use motion segmentation results to avoid invalid flow comparisons when operating at the edge of objects, and employ simple tracking of flow vectors to improve performance and detect discontinuities. We have demonstrated that the detected discontinuities can be used to detect occlusions between objects in the scene. The proposed algorithm has been compared to other commonly used optical flow algorithms and we have shown that for the task of extracting an object with a known velocity, the proposed algorithm performs very well.

REFERENCES

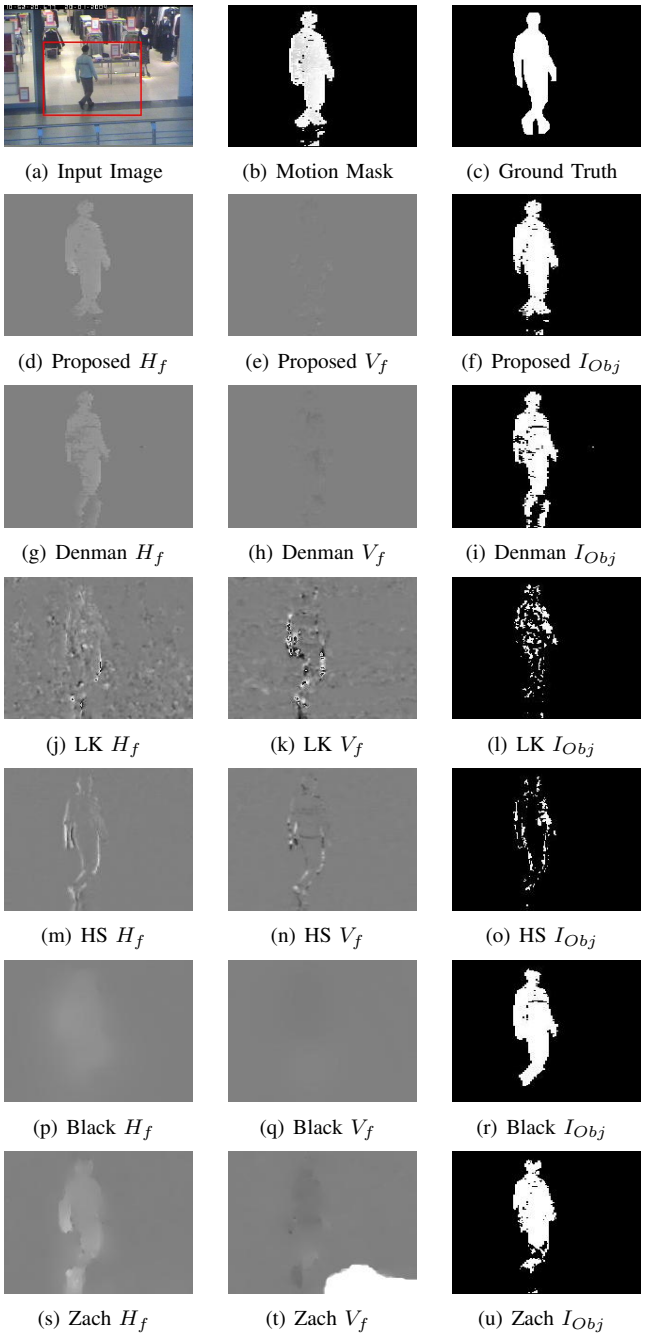
[1] B. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *7th International Joint Conference on Artificial Intelligence (IJCAI)*, 1981, pp. 674–679.

[2] B. K. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, pp. 185–203, 1981.

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

[4] M. Black and P. Anandan, "A framework for the robust estimation of optical flow," in *Fourth International Conference on Computer Vision*, 1993, pp. 231 – 236.

[2]Source code for [7] can be found at http://gpu4vision.icg.tugraz.at/index.php?content=downloads.php.



(a) Input Image (b) Motion Mask (c) Ground Truth

(d) Proposed H_f (e) Proposed V_f (f) Proposed I_{Obj}

(g) Denman H_f (h) Denman V_f (i) Denman I_{Obj}

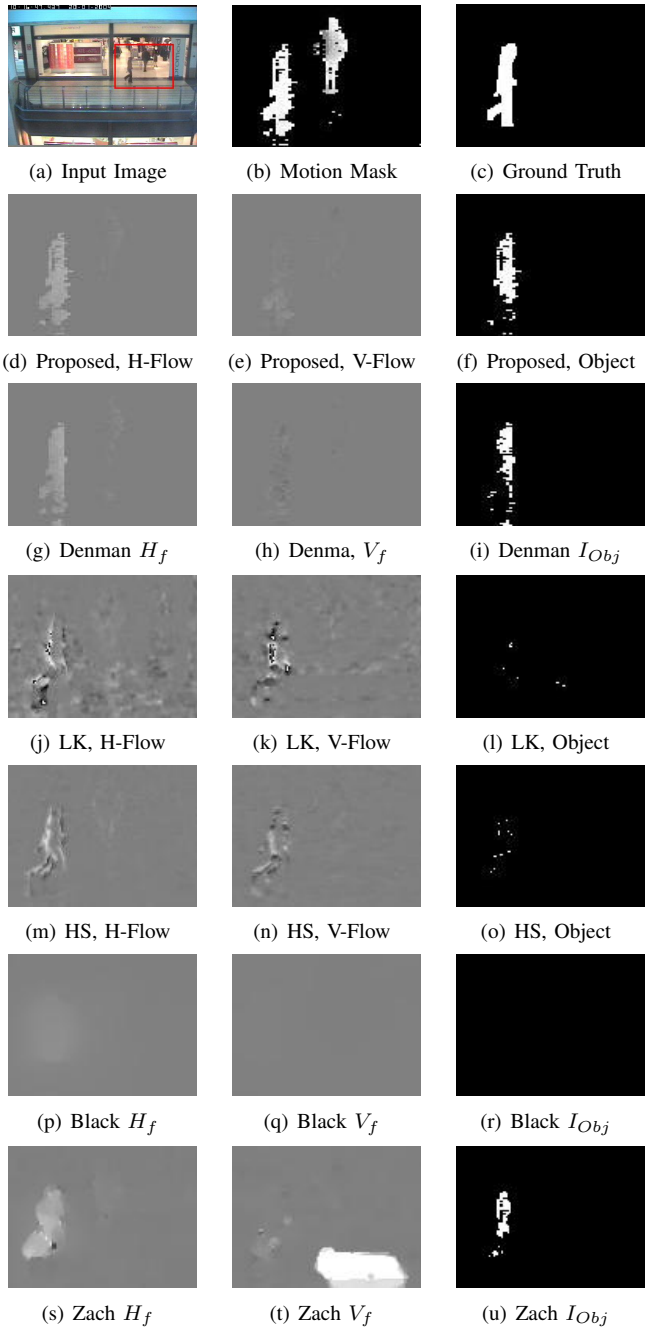
(j) LK H_f (k) LK V_f (l) LK I_{Obj}

(m) HS H_f (n) HS V_f (o) HS I_{Obj}

(p) Black H_f (q) Black V_f (r) Black I_{Obj}

(s) Zach H_f (t) Zach V_f (u) Zach I_{Obj}

Figure 4. Optical Flow Performance - CAVIAR Set WalkByShop1front (WBS1 front), Frame 1640



(a) Input Image (b) Motion Mask (c) Ground Truth

(d) Proposed, H-Flow (e) Proposed, V-Flow (f) Proposed, Object

(g) Denman H_f (h) Denma, V_f (i) Denman I_{Obj}

(j) LK, H-Flow (k) LK, V-Flow (l) LK, Object

(m) HS, H-Flow (n) HS, V-Flow (o) HS, Object

(p) Black H_f (q) Black V_f (r) Black I_{Obj}

(s) Zach H_f (t) Zach V_f (u) Zach I_{Obj}

Figure 5. Optical Flow Performance - CAVIAR Set OneStopNoEnter2front (OSNE2 front), Frame 1085

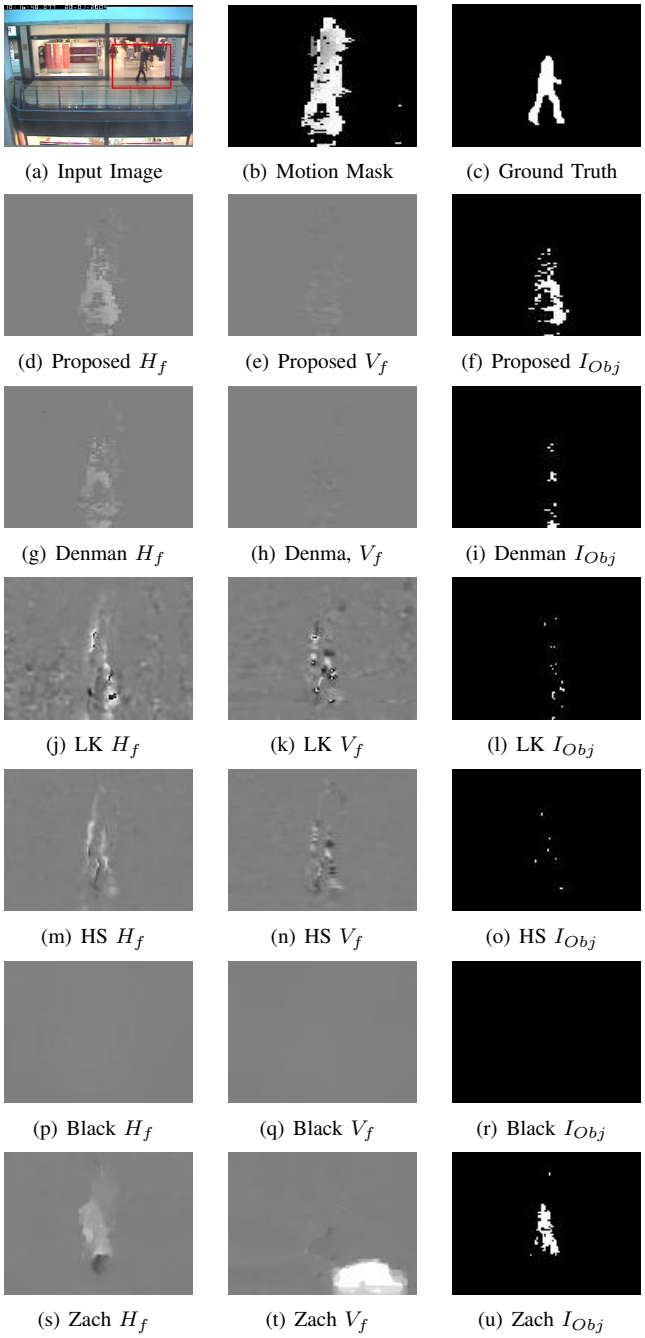
[5] S. Ince and J. Konrad, "Occlusion-aware optical flow estimation," *IEEE Transactions on Image Processing*, vol. 17, no. 8, pp. 1443–1451, 2008.

[6] C. Strecha and L. V. Gool, "Pde-based multi-view depth estimation," in *1st Int. Symp. 3D Data Processing Visualization and Transmission*, vol. 2, 2002, p. 416425.

[7] C. Zach, T. Pock, and H. Bischof, "A duality based approach for realtime tv-l1 optical flow," in *Pattern Recognition (Proc. DAGM)*, Heidelberg, Germany, 2007, pp. 214–223.

[8] M. Lucena, J. Fuertes, J. Gomez, N. de la Blanca, and A. Garrido, "Tracking from optical flow," in *Image and Signal Processing and Analysis, 2003. ISPA 2003. Proceedings of the 3rd International Symposium on, Vol.2, Iss., 18-20 Sept. 2003*, 2003, pp. 651– 655 Vol.2.

[9] N. Oshima, T. Saitoh, and R. Konishi, "Real time mean



(a) Input Image (b) Motion Mask (c) Ground Truth

(d) Proposed H_f (e) Proposed V_f (f) Proposed I_{Obj}

(g) Denman H_f (h) Denma, V_f (i) Denman I_{Obj}

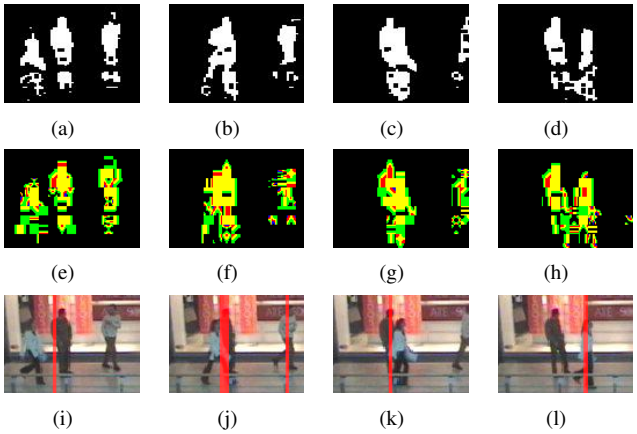
(j) LK H_f (k) LK V_f (l) LK I_{Obj}

(m) HS H_f (n) HS V_f (o) HS I_{Obj}

(p) Black H_f (q) Black V_f (r) Black I_{Obj}

(s) Zach H_f (t) Zach V_f (u) Zach I_{Obj}

Figure 6. Optical Flow Performance - CAVIAR Set OneStopNoEnter2front (OSNE2 front), Frame 1101

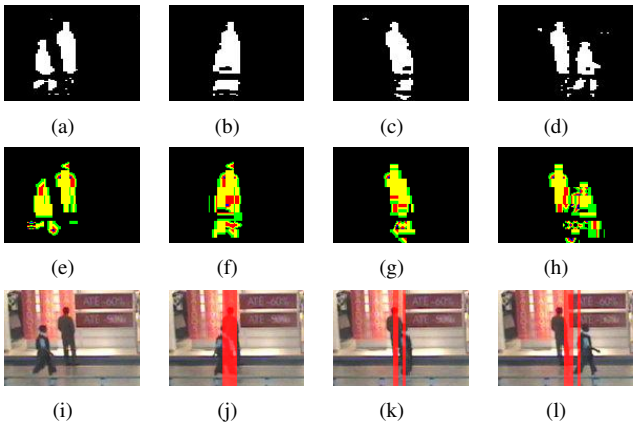


(a) (b) (c) (d)

(e) (f) (g) (h)

(i) (j) (k) (l)

Figure 7. Overlap Detection - Example 1



(a) (b) (c) (d)

(e) (f) (g) (h)

(i) (j) (k) (l)

Figure 8. Overlap Detection - Example 2

shift tracking using optical flow distribution," in *SICE-ICASE, 2006. International Joint Conference*, 2006, pp. 4316–4320.

[10] T. 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.

[11] I. Haritaoglu, D. Harwood, and L. Davis, "An appearance-based body model for multiple people tracking," in *15th International Conference on Pattern Recognition*, vol. 4, Barcelona, Spain, 2000, pp. 184–187.

[12] S. Denman, V. Chandran, and S. Sridharan, "An adaptive optical flow technique for person tracking systems," *Pattern Recognition Letters*, vol. 28, no. 10, pp. 1232–1239, 2007.

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

[14] C. Stauffer and W. Grimson, "Adaptive background mixture models for real-time tracking," in *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, vol. 2, 1999, p. 252 Vol. 2.

[15] R. Fisher, J. Santos-Victor, and J. Crowley, "Caviar: Context aware vision using image-based active recognition, (http://homepages.inf.ed.ac.uk/rbf/caviar/)," 2002, Last Accessed 23 Feb 2008.