DETECTING SHADOWS AND LOW-LYING OBJECTS IN INDOOR AND OUTDOOR SCENES USING HOMOGRAPHIES

Philip Kelly [*], **Paul Beardsley** [±], **Eddie Cooke** [*], **Noel O'Connor** [*], **Alan Smeaton** [*]

[*] Centre for Digital Video Processing, Adaptive Information Cluster, Dublin City University, Ireland
{kellyp, ej.cooke, oconnorn}@eeng.dcu.ie asmeaton@computing.dcu.ie
[±] Mitsubishi Electric Research Laboratories, 201 Broadway, Cambridge MA 02139, pab@merl.com

Keywords: Homography, Shadow Removal, Low-Lying Regions, Stereo, 2D Analysis

Abstract

Many computer vision applications apply background suppression techniques for the detection and segmentation of moving objects in a scene. While these algorithms tend to work well in controlled conditions they often fail when applied to unconstrained real-world environments. This paper describes a system that detects and removes erroneously segmented foreground regions that are close to a ground plane. These regions include shadows, changing background objects and other low-lying objects such as leaves and rubbish. The system uses a set-up of two or more cameras and requires no 3D reconstruction or depth analysis of the regions. Therefore, a strong camera calibration of the set-up is not necessary. A geometric constraint called a homography is exploited to determine if foreground points are on or above the ground plane. The system takes advantage of the fact that regions in images off the homography plane will not correspond after a homography transformation. Experimental results using real world scenes from a pedestrian tracking application illustrate the effectiveness of the proposed approach.

1 Introduction

Numerous computer vision applications depend on accurate detection and segmentation of moving objects from a background model as a first step in their algorithmic process. Many of the approaches implemented to achieve this are based on background suppression techniques that work well in controlled conditions. However, when these are applied to unconstrained real-world environments containing dynamic background conditions they often fail. Shadows, *ghosts* (introduced when objects that initially belonged to the background model move) and other small objects such as leaves or rubbish entering an outdoor scene can be incorrectly identified as valid foreground objects. While techniques exist to detect and remove shadows and ghosts by means of exploiting colour and motion information [7, 8, 9], this paper describes a method to remove not only ground plane shadows and ghosts but also other low-lying objects on a plane using a purely 2D geometrical approach.

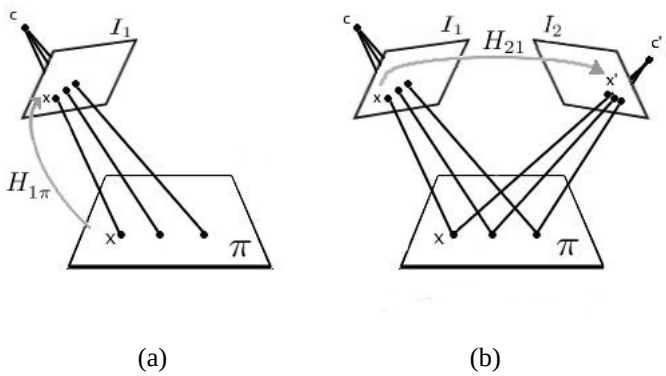
The technique described in this paper makes use of a geometric constraint called a *homography* between multiple views of the same scene taken from slightly different viewpoints. The homography is used to map points on the ground plane in one image to the corresponding ground plane points in another image of the same scene. Other published works using homographies include obstacle detection and avoidance [1, 2, 6] and road region extraction [12]. Although 3D depth information could be computed to determine shadows and low-lying regions across image pairs it is computationally expensive. Computing depth using stereo is of the order of $O(m * n * d)$, where m is the number of image pixels, n is the size of the correlation window, and d is the number of disparities searched [2]. Stereo-based homography on the other hand is computationally cheap. While it does not provide depth information, it does however provide enough information to determine whether a particular image point is on the ground plane or above it. Therefore, the algorithm described in this paper has two purposes: (i) it identifies and removes shadows and low-lying objects from segmented foreground regions and (ii) it can be used in a preprocessing phase to remove such objects before depth analysis and thereby improving the speed and reliability of the 3D reconstruction.

This paper is organized as follows: Section 2 presents the details of the developed algorithmic approach. Firstly, the homographic transformation and a robust estimation technique are described; we then illustrate how the background modelling is implemented and how both low-lying and rising low-lying regions are detected. In Section 3 we present experimental results from both a controlled indoor environment and a real world outdoor situation. Finally, Section 4 details conclusions and future work.

2 Algorithm Details

A homography, H, also referred to as a *projective transformation* or *collineation*, is an invertible mapping within 2D projective space, $\mathbb{P}^2$, such that three homogeneous points $\mathbf{x_1}$, $\mathbf{x_2}$ and $\mathbf{x_3}$ lie on the same line if and only if $H(\mathbf{x_1})$, $H(\mathbf{x_2})$ and $H(\mathbf{x_3})$ are collinear [10]. A 3×3 homography matrix can be used to describe both the relationships between real world planes and a camera's image plane and between two perspective views of a planar object.

Assuming a point, $\mathbf{X}$, on a planar surface, π, is projected to the



(a) (b)

Figure 1: (a) Homography $H_{1\pi}$ between a world plane and the planes image; (b) Homography H_{21} induced by a plane.

point $\mathbf{x}$ in the image plane I_1, a homography $H_{1\pi}$ exists from π to I_1 as shown in Figure 1(a). Taking a second camera and image, I_2, of the same plane a second distinct homography $H_{2\pi}$ exists from π to I_2. The composition of the two homographies, $H_{1\pi}$ and $H_{2\pi}$, results in a third homography [16], which exists from I_1 to I_2:

$$\mathbf{x}' \cong H_{2\pi} H_{1\pi}^{-1} \mathbf{x} \quad (1)$$

$$\mathbf{x}' \cong H_{21} \mathbf{x} \quad (2)$$

where $\cong$ denotes equality up to a scale factor.

This homography from I_1 to I_2, illustrated in Figure 1(b), is known as a *plane induced* homography as H_{21} depends on the position of the real world plane π. The effect of H_{21} is to match an image point $\mathbf{x} = (x, y, 1)^T$ on I_1 with its corresponding point $\mathbf{x}' = (x', y', 1)^T$ on I_2 iff both $\mathbf{x}$ and $\mathbf{x}'$ are images of the same point X on the plane π. In other words, Equation (2) is true if $\mathbf{x}$ and $\mathbf{x}'$ are images of the same point X that is on the plane π, otherwise Equation (2) fails. The farther the point $\mathbf{X}$ is off the homography plane the greater the disparity between $H_{21}\mathbf{x}$ and $\mathbf{x}$'s actual corresponding point in I_2.

2.1 Homography Estimation

The 3×3 homography matrix, H, has 9 entries, but it has only 8 degrees of freedom as it is a homogeneous matrix and therefore defined only up to scale. Each 2D homogeneous image point $\mathbf{x}_i = (x_i, y_i, 1)^T$ has two degrees of freedom corresponding to the x_i and y_i components. A point correspondence $\mathbf{x}_i \rightarrow \mathbf{x}'_i$ between the two projective planes gives two constraints on H, since for each point $\mathbf{x}_i$ on one projective plane the two degrees of freedom of the second point must correspond to the mapped point $H\mathbf{x}_i \cong (x'_i, y'_i, 1)^T$. Therefore at least four point correspondences are needed to constrain H fully. The point correspondences are found by detecting a calibration shape in an image and matching the found image points to real world points on the calibration shape. The calibration shape is placed at multiple positions on the ground plane in order to ensure a good distribution of corresponding points across the whole real world plane. This is necessary as one important source of errors is the uneven distribution of input points [6]. Work has been carried out on automatic extraction of feature points for planar homography estimation, see [15, 11, 4, 12] for more details. The normalised DLT algorithm [10] is used to compute a linear solution to the homography. The DLT algorithm generates a 2×9 matrix A_i for each point correspondence $\mathbf{x}_i \rightarrow \mathbf{x}'_i$, where

$$A_i = \begin{bmatrix} 0^T & -x_i^T & y'_i x_i^T \\ x_i^T & 0^T & -x'_i x_i^T \end{bmatrix} \quad (3)$$

If there are n point correspondences then the nA_i matrices are composed into a single $2n \times 9$ matrix A. A linear solution to $Ah = 0$ is found using the least-squares solution [10] to a homogeneous system of linear equations, where h is a 9-vector made up of the entries of the matrix H.

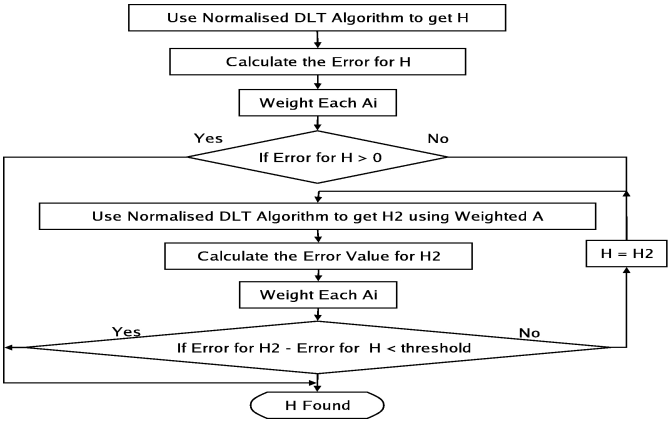


Figure 2: Iterative DLT flow diagram.

2.1.1 Robust Homography Estimation

Due to incorrect matches in corresponding feature points, called *outliers*, and also due to noise, errors arise in the homography estimation process. For a given point correspondence $\mathbf{x}_i \rightarrow \mathbf{x}'_i$ an error value, δ_i, can be calculated as the perpendicular distance between the real corresponding point to $\mathbf{x}_i$, namely $\mathbf{x}'_i$, and the corresponding point induced by H, namely $H\mathbf{x}_i$. The total error value, δ_H, for a given H can then be calculated as the sum of the errors that H induces on the point correspondences $\mathbf{x}_i \rightarrow \mathbf{x}'_i$. If there are more than four point correspondences, this error value can be improved by applying the normalised DLT algorithm iteratively. Figure 2 shows the iterative normalised DLT algorithm process flow diagram. The iterative process weights each set of the nA_i matrices that correspond to the n point correspondences to improve the estimation accuracy. The value of this weighting, α, is inversely proportional to the error:

$$\alpha A_i = A_i \cdot \frac{1}{\delta_i} \quad (4)$$

(a) (b) (c)

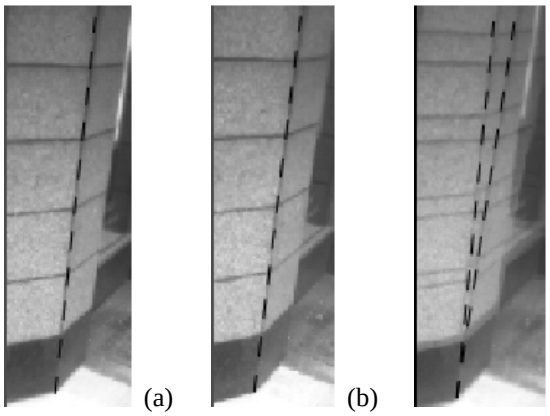
Figure 3: (a) Reference image, I_1; (b) Other image, I_2; (c) Transformed I_2 overlaid onto I_1.

As such if the point correspondence represented by A_i gives a large error, i.e. the point is an outlier, then A_i is scaled down. This means that the error induced by the homography for that point correspondence will also be very small, and therefore will be ignored to a large extent by the minimization algorithm. Alternatively if the point correspondence is found to fit well then A_i is scaled up. Finally once a linear solution to H is found, it is then enhanced using Powell minimization [13] to get a non-linear solution, this is necessary as linear methods generally suffer from more errors when compared to non-linear methods [6].

2.2 Background Modelling

Before extracting foreground regions, each of the reference images is transformed via H into the same image plane for comparison. Let I_1 represent the image plane and let $I_2 ... n$ be the $n-1$ other images of the same scene taken from different viewpoints. To transform each image, I_i, to the reference image I_1, Equation (2) is used, where H is the homography from image I_i to image I_1, see Figure 3 for an illustrative result. This is implemented to take advantage of the fact that two corresponding points on the ground plane would transform to the same point in the reference view, whereas points above or below the ground plane would not transform consistently. This can be seen from the zoomed section of the images in Figure 4 note the increasing disparity between the edge of the building, which is perpendicular to the ground plane, the farther the height above the ground plane.

After transforming the images an edge-based background subtraction technique [3] is employed to extract foreground edge pixels. The technique assumes a static background and camera and incorporates the background subtraction technique directly into the edge detection process, to produce a modified version of the standard Canny edge detector [5] which computes only those edges which differ from the static background. An edge-based approach is used as it provides the required input features that would be needed to generate a full 3D model which will be necessary in future work. Each background and foreground edge pixel has a set of data



(a) (b) (c)

Figure 4: (a) Zoomed section of reference image, I_1; (b) Zoomed section of other image, I_2; (c) Zoomed section of transformed I_2 overlaid onto I_1.

associated with it, this data is as follows:

1. A Gaussian for the pixel RGB value;
2. A Gaussian for the gradient magnitude, g^{mag}, which is the absolute gradient value of the RGB channel with the strongest gradient;
3. A Gaussian for the gradient direction, g^{dir}.

Using this technique the foreground edges are obtained in each of the n images. See Figures 7(a)-(d) and 9(a)-(d) for the results of the background subtraction technique in two different scenarios. The use of a background suppression technique can lead to a number of problems. The background may change due to two factors [7]:

1. Variations in the lighting conditions, for example, clouds covering the sun in outdoor scenes or lights turned on or off in indoor scenes.
2. Objects that modify their status from stopped to moving or vice versa.

Background suppression has an inherent trade-off between high responsiveness to changes and reliable background model computation which can lead to erroneous foreground region segmentation. An example of this is shadows cast by both foreground and background objects being detected as foreground objects. This is especially true in a dynamic real world environment where the occluded sun can suddenly emerge from behind clouds casting strong shadows. In addition background suppression can pick up *ghosts* as foreground regions. Ghosts are introduced when objects belonging to the background model move. When this occurs two new foreground regions are detected; the region where the object has moved to, and also the region where the object

was previously located in the background model. The second region is referred to as a ghost, since it does not correspond to any real moving object [7]. An example of a ghost can be found at the back right hand corner of Figure 9(d) and of a shadow in Figure 7(d), these are typical problems that occur with background suppression algorithms.

Work has been done on suppressing shadows and ghosts by means of exploiting colour information [7, 8, 9]. However, our approach aims to use a purely geometrical approach to detecting objects on or close to the ground plane. Ground plane objects, such as rubbish, cast shadows, and ghosts that are on the ground plane can therefore be eliminated as they will closely abide by the homography constraint set out in Equation (2). By using the homography information, the background subtraction technique can be supplemented in order to develop a more robust background subtraction algorithm.

2.3 Low-Lying Region Detection

Let I_1 be the reference image and let $I_2...n$ be $n-1$ other images of the same scene taken from different viewpoints that have been warped into the reference frame of I_1. Let the segmented foreground edge pixels in I_1 be $e_{11}, e_{12}...e_{1n}$, similarly let the foreground edge pixels in I_2 range from $e_{21}, e_{22}...e_{2m}$. ¿From the background edge subtraction technique described in Section 2.2, each foreground edge pixel e_{ri} has data associated with it. This data is used in conjunction with the homography information to localise low-lying regions from all detected foreground regions.

If a point e_{ri} actually *does* lie on the real world ground plane, then the edge data for the pixel at location i in each $I_2...n$ will closely match the edge data for the corresponding pixel in I_1. Alternatively, if a point e_{ri} lies above the real world ground plane, then the corresponding edge data in each $I_2...n$ should not be matched by the edge data for I_1 at the location i, and this discrepancy can be detected. However due to a number of reasons such as noise, image quantisation, uneven ground plane, lighting and incorrect homography estimation, this will not always be the case. A match to e_{ri} is therefore allowed to occur in a small local neighbourhood to e_{ri}, the size of this neighbourhood depends on both the position of the camera above the plane and also the distance above this plane that pixels should still be detected as low-lying, see Figure 10 for an illustration of this. To determine if an edge pixel e_{sj} is acceptable as a match to e_{ri}:

$$\sqrt{(g_{ri}^{dir} - g_{sj}^{dir})^2} < t_{seed} \tag{5}$$

where g_{ri}^{dir} is the edge gradient direction at e_{ri}, g_{sj}^{dir} is the edge gradient direction at e_{sj} and t_{seed} is a threshold value for the gradient directions of seed pixels. If a match to e_{ri} is found in *all* other reference images, then e_{ri} is labelled as a low-lying *seed pixel*, otherwise it is labelled as a pixel off the ground plane. By applying this algorithm with a relatively *low threshold* for t_{seed} reliable low-lying region seed pixels are obtained. Information on how to determine t_{seed} is provided in Section 3 and Figures 7(e) and 9(e) illustrate results for two different camera setups.

The generation of reliable seed edge pixels is integral to this technique, however, at this stage there are cases when spurious low-lying seed pixels are detected. The removal of these seed pixels is a two stage process, firstly the knowledge that ground plane regions should have clusters of seed pixels is applied. The number of seed and non-seed foreground pixels in the neighbourhood of each seed e_{ri} are computed. If e_{ri} does not have above a certain percentage, t_{perc}, of seed to non-seed foreground pixels then e_{ri} is reclassified as a non-seed pixel. However, the removal of seed pixels from one image implies that corresponding seed pixels in other images may no longer be valid. The second stage therefore is to check that each e_{ri} has a matching edge pixel in each other image, within the allowed neighbourhood, if it does not then e_{ri} is reclassified as a non-seed pixel. Information on how to determine t_{perc} is provided in Section 3.

Finally, each seed pixel is grown as follows. Each pixel j in the small local neighbourhood of each seed pixel e_{ri} is checked to see if it can be reclassified as a seed pixel. If j is a foreground pixel but not a seed or grown pixel then an attempt is made to find a match to pixel e_{cj} in *any* other reference image within the local neighbourhood to e_{ri}. Equation (5) is used with an increased threshold, t_{grow}, to determine if two edge pixels are a match. A match is allowed to occur in *any* other image instead of *every* other image, as in the case with obtaining seed pixels. The reason for this is that matches to all low-lying edge pixels will not be available in every image, due to occlusion induced by the different viewpoints of the cameras. The growing of seed pixels is done iteratively as initial seed pixels may be sparsely spread throughout the image. The number of iterations t_{iter}, is dependent on t_{seed}, since seed pixels become sparser as this threshold is lowered. By applying this algorithm with relatively *high thresholds* for t_{grow} reliable low-lying edge pixels are obtained. Information on how to determine both t_{grow} and t_{iter} is provided in Section 3 and Figures 7(f) and 9(f) illustrate results of low-lying pixels for two different camera setups.

2.4 Rising Low-Lying Region Detection

The low-lying edge pixels detected will consist of objects that should be ignored such as shadows on the ground plane, objects near to the ground plane such as leaves, paper, etc. However there will also be low-lying edge pixels that are part of objects that rise above the ground plane, these include pedestrians feet, parts of prams or bicycles, etc. These are *not* regions to ignore and so must be reclassified as foreground regions.

A modified version of the iterative connective components 4-neighbourhood algorithm [14] is applied to group the foreground edge pixels into regions. The algorithm is altered so that a new region, R, is grown according to gradient direction and pixel colour. A foreground edge pixel, e_{ri}, is *not* part of the region of one of its neighbours, e_{sj}, if either of the following equations are true:

$$\sqrt{(g_{ri}^{dir} - g_{sj}^{dir})^2} < t_{dir} \tag{6}$$

$$\sqrt{(g_{ri}^{mag} - g_{sj}^{mag})^2} < t_{mag} \tag{7}$$

where g_{ri}^{dir} and g_{ri}^{mag} are the edge gradient direction and magnitude at e_{ri} respectively, similarly g_{sj}^{dir} and g_{sj}^{mag} are the edge gradient direction and magnitude at e_{sj} respectively and both t_{dir} and t_{mag} are threshold values. The thresholds t_{dir} and t_{mag} are not critical, in our experiments t_{dir} and t_{mag} are set to 15 and 50 respectively. These equations force pixels with abrupt changes in direction or gradient to become new regions, it will however allow for gradual changes in both gradient direction, which will allow for curved regions, and gradient magnitude.

Once connected regions are formed the following statistics for each region, R, are obtained: the percentage of low-lying to non low-lying region pixels R_{perc}; the centre of gravity of the low-lying edge pixels ζ_{low}; and the centre of gravity of all the other foreground edge pixels ζ_{non}. We use these statistics to determine if a region is low-lying, rising or non low-lying. The thresholds t_{low} and t_{high}, used in the following rules, both represent percentages of low-lying to non low-lying region pixels in a region. These thresholds are not highly critical, in our experiments t_{low} is set to 25% and t_{high} is set to 75%. For each R that has at least one low-lying region edge pixel the following is applied:

1. If $R_{perc} < t_{low}$; R is classified as a false positive low-lying region and every foreground edge pixel in R is reclassified as non low-lying region pixels.

2. If $R_{perc} > t_{high}$; R is classified as a true positive low-lying region and every foreground edge pixel in R is reclassified as low-lying region pixels.

3. If $R_{perc} > t_{low}$ and $R_{perc} < t_{high}$; then the relationship between the orientation of ζ_{low} to ζ_{non} is examined. This information is used to indicate if the region is a low-lying region that is rising above the ground. If the ground plane occurs at the bottom of the image and object rises above the ground to the top of the image then: the y value of ζ_{non} should be above the y value ζ_{low}; the absolute difference between the y values of ζ_{non} and ζ_{high} should be significantly larger than the absolute difference between the x values of ζ_{non} and ζ_{high}. If this is true then every foreground edge pixel in R is reclassified as rising low-lying region pixels.

By applying this technique, spurious low-lying regions will be eliminated, real low-lying regions will have missing sections filled in and rising low-lying regions will be reclassified as relevant foreground regions. See Figures 7(f)-(g) and 9(f)-(g), note how the problems of the shadow in Figure 7(d) and ghost at the back right hand corner in Figure 9(d) are eliminated. This method has also the added advantage that it becomes possible to tell if part of an object is close to the ground plane but the object rises above it, or if the object is never near the ground plane, such as a persons head and shoulders when their feet are not visible. This information will be valuable in our future work.

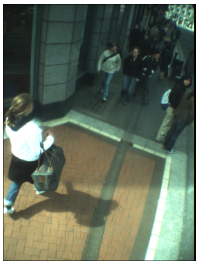
3 Experimental Results

The crux of this work is to obtain good low-lying region pixels. This means finding as many correctly identified pixels while minimizing the number of false positives for a given setup. The technique defined in this paper is therefore dependent on four major thresholds values, namely; t_{seed} and t_{perc} which are needed to obtain good seed pixels, and t_{grow} and t_{iter} which are used to constrain the growth of the seed pixels. In order to obtain the threshold values the system is put through a training stage. This training involves using manually masked input images, such as the one shown in Figure 5, to obtain a ground truth to allow the selection of the best thresholds for a given setup.

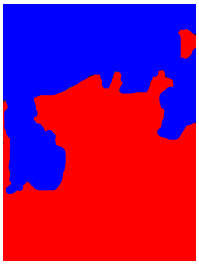
For a given threshold configuration, tc_i, the low-lying edge pixels are found in a single image via the defined algorithm. These pixels are then compared to the masked image and the number of correctly identified low-lying pixels, c_i, and the number of false positives, fp_i, are determined for each tc_i. Various tc_i are tested in a coarse to fine manner in order to obtain an optimal threshold configuration. A graph such as Figure 6(a) is the result of this process, it shows the values of c_i and fp_i for a number of configurations for a given image. The graph has been sorted using fp_i values. Analysis of this graph indicates that the value for t_{seed} is the dominating factor for both c_i and fp_i and it can be said that *in general* the graph's x-axis is mirrored by the value of t_{seed}, which ranges from 0.25 to 2.5 as the x-axis diverges from the origin. To determine how tc_i compares to the other threshold configurations tested, tc_i must be benchmarked against them. A *suitability value*, sv_i, is associated with tc_i which takes into account both how well the configuration finds accurate low-lying pixels and also how well it controls the generation of false positives in a certain image.

$$sv_i = \underbrace{\frac{100 \times n \times c_i}{\sum_{x=1}^{n} c_x}}_{A} - \underbrace{\frac{100 \times n \times fp_i}{\sum_{x=1}^{n} fp_x}}_{B} \tag{8}$$

The first part of Equation (8), A, represents the percentage of correct low-lying pixels found by tc_i to the average number found for all configurations for that image; part B of the



(a)



(b)

Figure 5: (a) Original foreground image; (b) Masked foreground image (Red: low-lying regions; Blue: non low-lying regions).

equation is similar to the first but calculates a percentage for false positives. Both A and B are given an equal weighting as we are interested in the solution that returns the most number of accurate low-lying pixels with the least number of false positives. If a threshold setup is required for higher precision rather than recall then the equation can be modified simply by weighting A at a higher value relative to B. The best tc_i is found by maximizing the value for sv_i. The graph in Figure 6(b) shows sv_i for the configurations sorted in the same order as Figure 6(a). Figure 6(c) illustrates c_i, fp_i and sv_i in the same coordinate system. It can be seen that the graph for sv_i mirrors closely that of c_i when fp_i is low, but as the number for fp_i becomes greater, sv_i rapidly declines. The region where sv_i peaks defines the boundaries of the area of interest of the coarse iteration process. A fine iteration can then be applied between these boundaries to determine the optimal threshold setup. This process is repeated for a number of images, we implement it for 10 images of varying conditions, and the tc_i with the highest sv_i averaged over the number of frames is the configuration used in a given setup.

We define two different setups; setup one consists of three cameras placed approximately 4 metres above the ground plane monitoring real world conditions, and the second setup consists of three cameras at a height of 1.5 metres above the ground plane monitoring indoor scenarios. The cameras are in a $45° - 45° - 90°$ triangular configuration with a baseline of approximately 10cm between each camera and a camera focal length of 3.8mm. The image resolutions are 640×480 and 320×240 respectively. For setup one it was found that a threshold configuration of t_{seed} = 1, t_{perc} = 33, t_{grow} = 10 and t_{iter} = 75 works best and for setup two a threshold configuration of t_{seed} = 2, t_{perc} = 25, t_{grow} = 5 and t_{iter} = 10 was used. Both setups used a neighbourhood of 1 vertical and 1 horizontal pixel and this can be increased to obtain pixels that lie at a greater height from the ground plane. The difference in the thresholds found for these two setups points to the fact that setup two was in a controlled environment with little clutter, background activity and consistent lighting conditions. This meant that the threshold on seeds and growth could be relaxed and therefore less iterations on growth were necessary. See Figures 7, 8 and 9 for detection results from both setups.

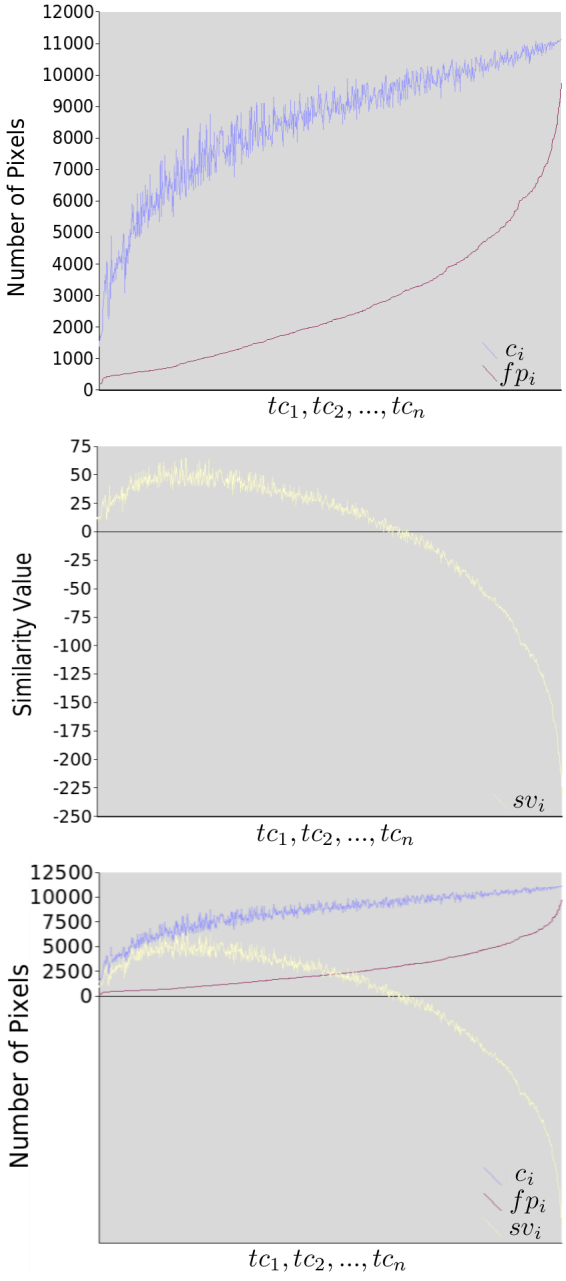


Figure 6: Graphs for a number of different threshold configurations (a) Graph of c_i (in blue) and fp_i (in red) (b) Graph of sv_i (c) Graph of c_i (in blue), fp_i (in red) and scaled up sv_i (in yellow).

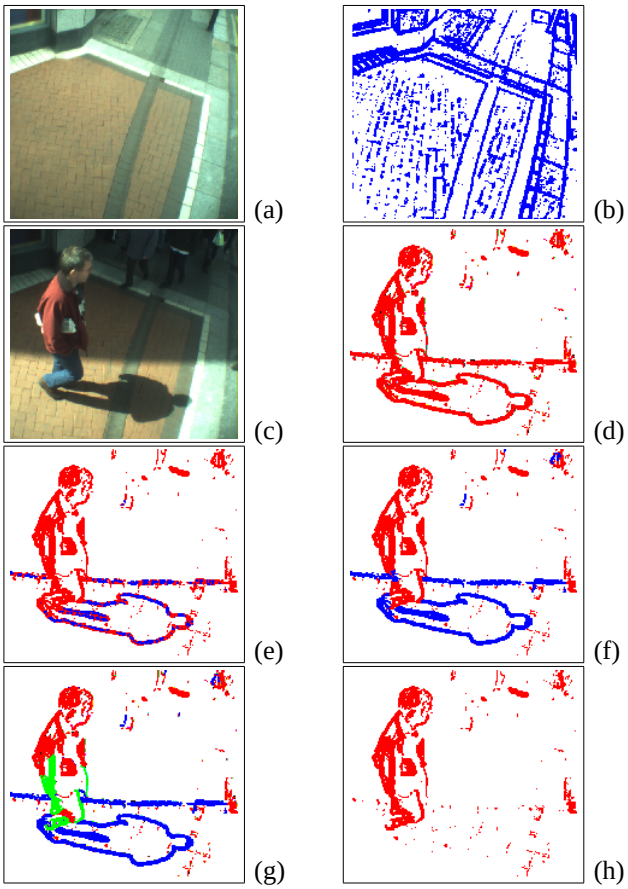


Figure 7: Results for setup one (a) Background image; (b) Background model edges points; (c) Foreground image; (d) Segmented foreground edge points; (e) Seed points (Blue points); (f) Grown seed points (Blue points); (g) Rising low-lying regions (Green points); (h) Only non low-lying regions.

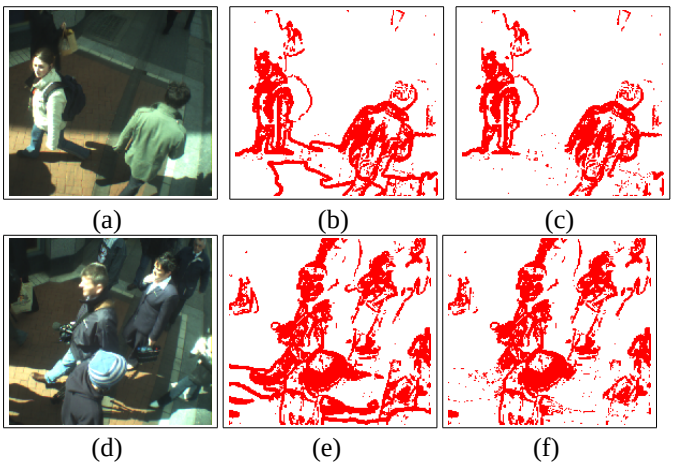


Figure 8: Results for setup one; (a) and (d) Foreground image; (b) and (e) Segmented foreground edge points; (c) and (f) Only non low-lying regions.

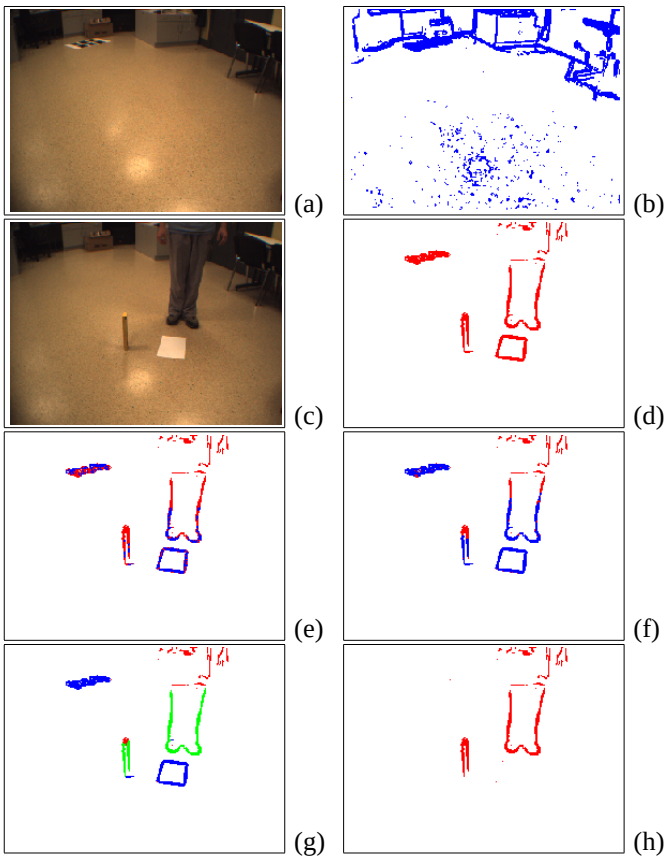


Figure 9: Results for setup two (a) Background image; (b) Background model edges points; (c) Foreground image; (d) Segmented foreground edge points; (e) Seed points (Blue points); (f) Grown seed points (Blue points); (g) Rising low-lying regions (Green points); (h) Only non low-lying regions.

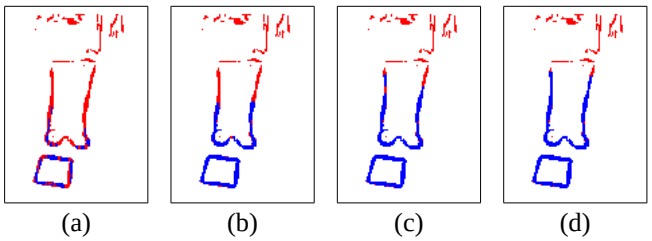


Figure 10: Effects of a changing neighbourhood size (Results shown are after growing Seed Edge Points), the neighbourhood region consists of an offset of ± 1 Vertical pixel and (a) ± 0 Horizontal pixels; (b) ± 1 Horizontal pixel (we use this size neighbourhood for the two setups); (c) ± 2 Horizontal pixels; (d) ± 3 Horizontal pixels.

4 Conclusions and Future Work

Many computer vision applications require background suppression techniques for the detection and segmentation of moving objects in a scene. While these algorithms tend to work well in controlled conditions they often fail when applied to unconstrained real-world environments. This paper described a system that successfully detects and removes erroneously segmented foreground regions in both constrained and unconstrained scenarios. The algorithm accurately detects regions such as shadows, ghosts and other low-lying objects. The approach avoids a costly computation of 3D depth values and instead uses an easily computed 2D geometrical constraint called a homography defined between multiple images of the same planar scene. Experimental results are presented that indicate the correctness of the approach for both an indoor scenario and the outdoor environment of a pedestrian tracking application.

The long term goal of this research is to develop a multiple camera system that can robustly track pedestrians in 3D. In that sense, the algorithm described may be regarded as a 2D preprocessing step designed to remove erroneous foreground information that would otherwise be tracked. Future work entails computing a full multi-camera calibration allowing the computation of fundamental matrices between camera pairs. The derived epipolar geometry will enable the generation of stereo correspondences within the images and hence 3D tracking. The creation of virtual views would also be possible and therefore allow users to view the scene from a chosen pedestrian's point of view.

Acknowledgements

This material is based on works supported by Science Foundation Ireland under Grant No. 03/IN.3/I361.

References

[1] R. Alix, F. Le Coat, and D. Aubert. Flat world homography for non-flat world on-road obstacle detection. In *Proceedings of the IEEE Symposium on Intelligent Vehicles*, pages 310–315, 2003.

[2] P. H. Batavia and S. Singh. Obstacle detection using adaptive color segmentation and color stereo homography. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 705–710, 2001.

[3] P. Beardsley and E. Bourrat. Wheelchair detection using stereo vision. Technical report, MERL, August 2002.

[4] B. Boufama and D. O'Connell. Region segmentation and matching in stereo images. In *Proceedings of the 16th International Conference on Pattern Recognition*, pages 631–634, 2002.

[5] J. Canny. A computational approach to edge detection. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 679–698, 1986.

[6] Y. H. Chow and R. Chung. Obstacle avoidance of legged robot without 3d reconstruction of the surroundings. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2316–2321, 2000.

[7] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati. Detecting objects, shadows and ghosts in video streams by exploiting color and motion information. In *11th International Conference on Image Analysis and Processing*, pages 360–365, 2001.

[8] R. Cucchiara, C. Grana, M. Piccardi, A. Pratii, and S. Sirotti. Improving shadow suppression in moving object detection with hsv color information. In *Proceedings of IEEE Intelligent Transportation System Conference*, pages 334–339, 2001.

[9] I. Haritaoglu, D. Harwood, and L. S. Davis. W4: Real-time surveillance of people and their activities. In *IEEE Trans. on Pattern Analysis and Machine Intelligence*, number 8, pages 809–830, 2000.

[10] R.I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision Second Edition*. Cambridge University Press, 2003.

[11] K. Okuma, J. J. Little, and D. G. Lowe. Automatic rectification of long image sequences. In *Asian Conference on Computer Vision*, 2004.

[12] M. Okutomi and S. Noguchi. Extraction of road region using stereo images. In *Proceedings of the 14th International Conference on Pattern Recognition*, pages 853–856, 1998.

[13] M. J. D. Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The Computer Journal*, 1964.

[14] M. Sonka, V. Hlavac, and R. Boyle. *Image Processing, Analysis and Machine Vision, Second Edition*. PWS Publishing, 1999.

[15] E. Vincent and R. Laganiere. Detecting planar homographies in an image pair. In *Proceedings of the 2nd International Symposium on Image and Signal Processing and Analysis*, pages 182–187, 2001.

[16] Q.-B. Zhang, H.-X. Wang, and S. Wei. A new algorithm for 3d projective reconstruction based on infinite homography. In *International Conference on Machine Learning and Cybernetics*, pages 2882–2886, 2003.