

Kernel-based object tracking using adaptive feature selection

Vladimir Nedovic, Martijn Liem, Maarten Corzilius and Mark Smids

Informatics Institute

University of Amsterdam

{vnedovic, mliem, mcorzili, msmids}@science.uva.nl

Abstract One of the major parts of the multimedia research is object localisation and object tracking. In this paper we try to create a surveillance system that uses object tracking and background subtraction. The final system can keep track of a person walking through a room and detect if something is left behind. For the tracker, adaptive feature selection is used which means that the tracker selects the best image feature it can find to track the object all by itself. For this purpose we implemented several image features. The background subtractor works using an adaptive background and motion detection system.

1. Introduction

Tracking is an important and challenging task within the field of computer science. It can be applied in a great number of different applications from various fields.

In surveillance, tracking can be used to detect and follow persons. Various types of behaviour can be detected. Fighting people or robberies can be detected, and appropriate actions can be taken by warning security personnel or police officers. Another surveillance application is the detection of suspicious objects, which might contain bombs or other dangerous contents.

In traffic control tracking can be used to detect vehicles on the road. This information can then be used in various ways. Detection of illegal conduct, such as speeding, changing lanes where it is not allowed or drunk driving is possible. Traffic flows can be monitored and used to steer traffic lights in the most efficient way. Tracking from within a car can be used to facilitate automatic driving systems.

In sport matches tracking can be used to track players on the field. This can lead to interesting trajectories and statistics of players performance. This information can be used to improve television broadcasts. Also useful for broadcasting is the tracking of the publicity boards along the edges of the field in many sport matches. It is commercially interesting to have these boards show localized publicity. Using tracking, the exact location of the

boards can be detected which creates the possibility to replace boards with different images.

In this paper we study various computational methods used to track objects in video sequences. We will focus our efforts on a mean-shift tracker, with adaptive feature selection. This tracker will be applied in a surveillance video setting. We will be using video material from the CAVIAR project, which is freely available online [10]. These videos contain manufactured surveillance material, in which people perform various activities, such as walking around, fighting and running.

Our tracker will have to be robust. Robust against changes in appearance, such as changes in colour, changes in light conditions and changes in shape or size. And robust against occlusion, which occurs when the tracked object moves behind another object, making the tracked object partially or even completely invisible. Furthermore we want the tracking to be performed in real-time.

To make optimal use of the results from our tracker, we will combine this system with additional image processing. A background subtraction component will be made, which will be used to detect foreign objects. Another component will be made to remove lens distortion and perspective from the video frames, so that a flat overview of the world is obtained. Detected objects and trajectories obtained from the tracker will be shown in this overview.

The combined system will track persons and detect suspicious objects. It will show all results in an overview image of the world.

2. Background and related work

The last decade a lot of work is presented dealing with tracking of objects in video. Various number of tracking methods are applied. One simple and popular method is tracking objects using template matching [11,12]. In this method the object is selected by drawing a bounding box and this area then acts as a model. In the next frames of the video we seek the area in the video frame which matches best given the model using different kind of features

[13,14,15]. Since this method uses an exhaustive search, a limitation of the search area is needed. If the objects are moving too fast in between frames the tracker might lose the objects. Tracking using general template matching is not robust to intensity changes, shape changes and (partial) occlusion, although attempts are made to overcome these problems [12,14].

A more popular robust tracking method is the mean shift method [4,16], which we also used for our project. We explain this method in more detail in chapter 3. As in [4] we also use an adaptive feature selection which selects and combines a number of top-ranked features at each frame. Features could be colour space models, wavelets or all kinds of filters.

For the construction of the trajectory model we have to deal with transformation of video frames. A related work that closely matches this part of our project is [6]. The construction of a homography is explained here, which we also used to obtain a top-view of the room. [6] also presents the combination of a tracker with this homography: a trajectory can be plotted on the model of the scene, which is exactly one of the goals in our project. An extension to obtain trajectory models in scenes where we have moving camera's is presented in [17,18] where the authors used different techniques to automatically construct a panorama from a set of images. This panorama could be used as trajectory model when we deal with camera movement, although it is not included in our project since the dataset of movies does not contain any camera movement.

Finally some tracking mechanisms also work with background subtraction [19]. In this case a certain frame of the video is considered as background (frame that does not contain moving objects). For every next frame the difference (in pixel values) between the images is taken. In this way new objects appearing in the scene will be visible and the background will be removed (black). We do not use background subtraction for tracking but we use it in combination with our tracker to detect if people are leaving something behind, which might be a suspicious event. In Chapter 7 we will describe our approach, with respect to background subtraction, in more detail.

3. Image Features

In order for our mean-shift system with adaptive feature selection to work properly, we will need to supply this system with a substantial number of different features. In this section we will discuss the

used features. These features can be separated in two main sections, colour based features and texture based features.

3.1 Colour based features

We have used the three different channels from three different colour models, for a total of 9 colour based features. The three colour models we used are RGB, rgb and HSV [20].

The RGB model is the colour model in which the frames are supplied. The R, G and B features respectively represent the amount of red, green and blue in the represented colour. Because these values are coming directly from the image, without transformation, any changes in the image are directly reflected in these values. Thus, a change in lighting intensity, colour or direction causes the same colour in the real world to get a different RGB representation. In a tracker this means that between two frames the tracked object can change in appearance considerably, which might cause problems in tracking.

The rgb colour model consists of the r, g and b features. These are the normalized colours, obtained by dividing the R, G and B values by their total sum.

$$r = \frac{R}{(R + G + B)} \quad (1)$$

$$g = \frac{G}{(R + G + B)} \quad (2)$$

$$b = \frac{B}{(R + G + B)} \quad (3)$$

Because the r, g and b values are calculated as the ratio of the R, G and B values, the luminance is factored out of the system. This has the effect that the rgb features are independent of lighting intensity and direction. This colour model is however still dependent on highlights and the lighting colour.

The HSV colour model is based on human colour perception. The features H, S and V represent respectively the hue, saturation and value of a colour. H is the colour aspect, S is the amount of white in the colour and V can be seen as the brightness of a colour, from dim to bright.

$$I = \frac{(R + G + B)}{3} \quad (4)$$

$$H = \arctan\left(\frac{\sqrt{3}(G - B)}{(R - G) + (R - B)}\right) \quad (5)$$

$$S = 1 - 3 \min(r, g, b) \quad (6)$$

In this colour model the different features have different dependencies. V is dependent on lighting direction, intensity, and colour. S is dependent on highlights and the lighting colour. H is only dependent on the lighting colour. Compared to the previous two models, the H value in this model adds independence of highlights.

3.2 Texture based features

The textures features we used are co-occurrence, Gabor filters and wavelet packets. These methods are aimed at finding texture within images. Since this can usually not be done using simple transformations, as with the colour models, these methods are more computationally expensive.

The co-occurrence features are calculated by counting the number of occurrences of two colours in some specific relation within the frame. Figure 1 shows an example of the calculation of the co-occurrence matrix in horizontal (left to right) direction. Figures 1a and 1b show two different images. Figure 1c shows the colour distribution histogram for both images, which is the same. Based on colour distribution these two images cannot be distinguished. Figures 1d and 1e show the co-occurrence matrices for respectively the first and the second image. These two matrices are clearly different, and are therefore very useful for distinguishing the two images.

For our tracker we have used co-occurrence features in four directions; horizontal, diagonal (top left to bottom right), vertical and diagonal (top right and bottom left).

Both Gabor filters and wavelet packets work under the assumption that the energy distribution in the frequency domain of (part of) an image identifies a texture. By filtering on specific frequencies or

decomposing the image in a number of frequency sub bands, the energy signatures of different textures will be different. [21]

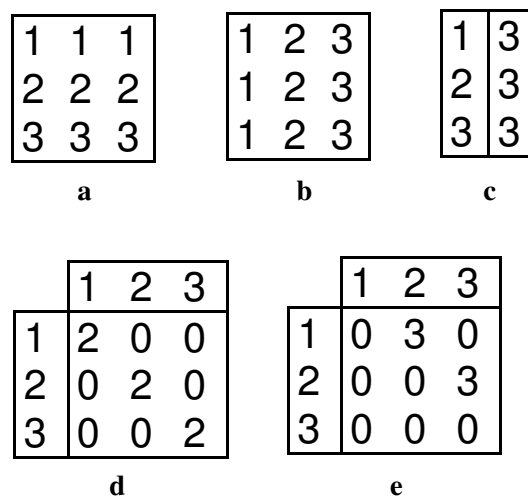


Figure 1:
 a. A three-colour image
 b. A three-colour image
 c. Colour distribution histogram
 d. Co-occurrence matrix for first image
 e. Co-occurrence matrix for second image

Gabor filters are Gaussians modulated by complex sinusoids. For our project we used the implementation by Ahmed Poursaberi, available online [22], which uses the following equation to calculate the Gabors:

$$G(x, y, \theta, f) = \exp\left(-\frac{1}{2} \left\{ \left(\frac{x'}{sx'}\right)^2 + \left(\frac{y'}{sy'}\right)^2 \right\}\right) * \cos(2 * \pi * f * x') \quad (7)$$

with

$$x' = x * \cos(\theta) + y * \sin(\theta) \quad (8)$$

$$y' = y * \cos(\theta) - x * \sin(\theta) \quad (9)$$

The results of these calculations are filters as shown in figure 2. These filters are convolved with the original image to produce the filtered results. Since there are a lot of possible combinations of x, y, theta and f, which can all be applied to any of the nine colour features available, it is not realistic to use all of these combinations as possible features.

We have opted to use only three Gabor filter features, thus three combinations of variables. To find the best variable settings to use in our application, we have performed test runs on the data set, using the feature selection functionality which will be described at another point in this report. From these test runs the settings from three of the better performing features were selected to be used in the final program. Even though these features are not the best for every possible situation, they are certainly among the Gabor filters best suited for our current needs.

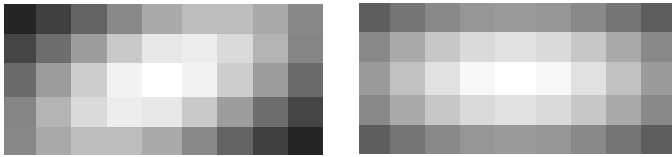


Figure 2: Two examples of the Gabor filters used for our project.

Wavelet packets are filter banks, used for decomposing an image into its frequency sub bands. A ‘standard’ wavelet transform is based on a dyadic sub band structure, or octave band decomposition. Research shows however that texture features are most prevalent in the intermediate frequency bands. The wavelet packet transform is a wavelet transform with sub band decompositions that are not restricted to the dyadic type. [21] For our project we have used wavelet packets in the horizontal, vertical and diagonal directions. The result for a single wavelet sub band in these three directions is shown in figure 3.

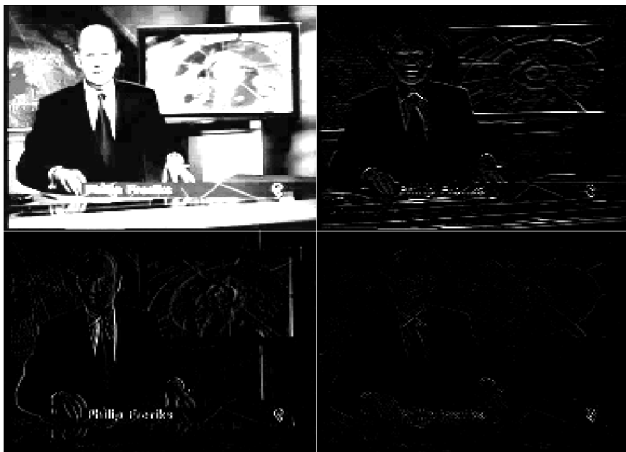


Figure 3: Wavelet transform on an image (left upper), in horizontal (left lower), vertical (right upper) and diagonal (right lower) directions.

4. Mean-Shift tracker

4.1 The tracker

In mean shift trackers, objects of interest are characterized by the probability density functions (pdfs) of their colour or texture features. By masking distributions with a monotonically decreasing kernel, a spatially-smooth similarity function between the original object and the target candidates can be defined; mean shift iterations can then use a local maximum of this similarity function as an indicator of the direction of target’s movement. The kernel is given by

$$K_E(x) = \frac{1}{2} c_d^{-1} (d+2) (1-\|x\|^2) \quad (10)$$

(if $\|x\| < 1$ and 0 otherwise),

where c_d is the volume of the unit d-dimensional sphere and $\|x\|$ indicates a vector norm (i.e. the kernel assigns larger weights to pixels closer to the object’s centre). Since we were dealing with a two-dimensional image space, our kernel function was of the form

$$K_E(x) = \frac{2}{\pi} (1-\|x\|^2) \quad (11)$$

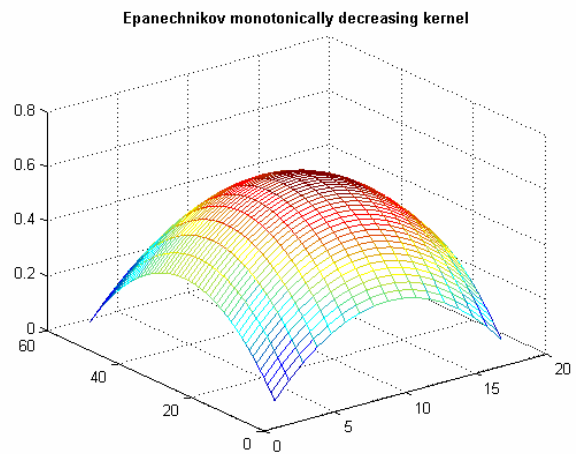


Figure 4: Epanechnikov kernel used to obtain a smooth similarity function used in tracking

The rationale for using a kernel to assign smaller weights to pixels farther from the centre is that those pixels are the least reliable, since they are the ones most affected by occlusion or interference from the background. A kernel with Epanechnikov profile was essential for the derivation of the smooth

similarity function between the distributions, since its derivative is constant; thus the kernel masking lead to a function suitable for gradient optimization, which gave us the direction of the target's movement. The search for the matching target candidate in that case is restricted to a much smaller area and therefore it is much faster than the exhaustive search.

The similarity between distributions is expressed in terms of Bhattacharyya coefficient, given by

$$\rho [p(\mathbf{y}), q] = \int \sqrt{p_z(\mathbf{y})q_z} dz \quad (12)$$

This similarity metric is related to Bayes classification error, which in turn is directly related to the similarity of distributions. It is argued to be much more suitable than many more commonly employed techniques, such as histogram intersection [1,2].

In order to maximize Bhattacharyya coefficient we needed to maximize a density estimate, which was equivalent to a histogram weighted by a kernel. Details of the mean shift procedure are given in [1-3].

4.2 Adaptive Feature Selection

As indicated above, in mean-shift tracking, objects are characterized by some feature's probability distribution function. The degree to which the object is discriminated from the background heavily depends on the feature(s) selected for the tracking purpose. However, most available tracking applications use a fixed set of features which are determined a priori (i.e. offline). The work of Collins and Liu [4] proposes a simple feature selection mechanism to choose discriminative features while tracking (i.e. online). In this approach, multiple feature spaces are constantly being considered and their discriminative power evaluated independently based on a simple two-class variance ratio measure; then the best feature(s) are selected for tracking. (The features are considered independently in order to reduce the search space for feature subsets from 2^n - since selection should be performed online, the speed is favoured over optimality.)

In the proposed feature selection method, the selection issue is seen as a two-class discrimination problem, where the two classes are object and background. Histograms of features for both classes are computed as estimates of their distributions; the histograms are then used to obtain a log-likelihood

ratio of those distributions. The latter is done through the following formula:

$$L(i) = \log \frac{\max \{p(i), \delta\}}{\max \{q(i), \delta\}} \quad (13)$$

where $p(i)$ and $q(i)$ are discrete probability distribution estimates for object and background, obtained by normalizing the respective histograms by the number of elements in it; i represents a particular histogram bin and δ is a small value that prevents division by zero. This nonlinear function assigns positive values to feature values corresponding to the object, and negative ones to those corresponding to the background. When mapped back into the image space, the likelihood image of discriminative features shows the object as very distinct from the background; the distribution of values in this image is then used as an input to the variance ratio (Figure 5).

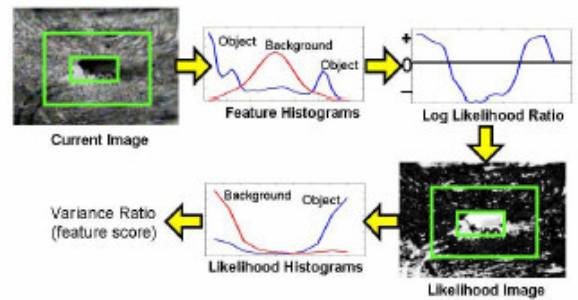


Figure 5: Feature selection process: feature histograms of object and background are used to obtain a log likelihood ratio which is mapped into the intensity values of a likelihood image; the distribution of values in this image is used as an input to the variance ratio score

Finally, a variance ratio of likelihood distributions is computed. A simple variant of the Fisher's criterion used in Linear Discriminant Analysis (LDA), called Augmented Variance Ratio (AVR), is used as a measure of class separability: it is the ratio of the between-class variance to the within-class variance, augmented by the difference between the class' means:

$$AVR(F) = \frac{Var(S_F)}{\frac{1}{C} \sum_{i=1..C} \min_{i \neq j} (|mean_i(S_F) - mean_j(S_F)|)}$$

where C is the number of classes and S_F is a feature being considered. Since the measure is applied to the distribution of log-likelihood values, we have that C

$= 2$ and $S_F = L$. In that case, the formula is of the following form:

$$AVR(L; p, q) \equiv \frac{2 \cdot |\mu_p - \mu_q| \cdot \text{var}(L; (p + q))}{[\text{var}(L; p) + \text{var}(L; q)]} \quad (14)$$

where the numerator represents the variance of L over both object and background (augmented by the difference between class means), and the denominator the sum of the variances for object and background taken separately. The original features are ranked according to their AVR score and top N features are used for tracking the object.

5. Implementation details

In our project, we had to expand on the existing work in which mean-shift tracking and most of the adaptive feature selection have already been implemented. Since we were studying mean-shift procedure in detail in previous courses, we chose not to present its details in this work – they can be found in one of our earlier project reports, namely in [3]. In case of feature selection, however, we had to familiarize ourselves with the underlying theory and the existing code; in addition, our contributions were made at multiple places. For these reasons, in the following section we give the overview of the whole feature selection process, as well as of the details referring to our own implementation decisions.

Feature Selection

As already mentioned, for feature selection we decided to use 3 colour spaces, as well as various texture features; for colour, we chose to work with RGB, rgb and HSV, and for texture we had 3 Gabor filters with various orientations, co-occurrence matrices, and 3 first high-frequency nodes obtained by wavelet packet decomposition. At the start of the process, the user can select the object he/she wants to track by specifying the vertices of the bounding rectangle, as shown below:



Figure 6: Selection window defines the object region, and it is expanded on each side by 30 pixels to form the background region

The selection defines a foreground window, from which the object is to be extracted; the region encompassing the foreground window, together with a 30 pixels-wide surrounding area, constitutes the background. Once we have these parameters, the whole process can start. It roughly consists of the following sequence:

1. *Calculating features:* all the features are extracted and processed to yield images (containing intensity values, filter coefficients, etc. - see Figures 1 and 2 in the Appendix), from which probability distributions can be estimated in the form of histograms. The histograms are weighted by corresponding Epanechnikov kernels described above. Since the object region is of rectangular shape and thus contains lots of background pixels, the variance within that region is very high, which results in a low AVR score (i.e. the better the segmentation within the foreground region, the lower the AVR score); for that reason, we chose to reduce the foreground-kernel's size by 30% from each side. In case of background, we use a kernel having window size, but discard contribution from object pixels.

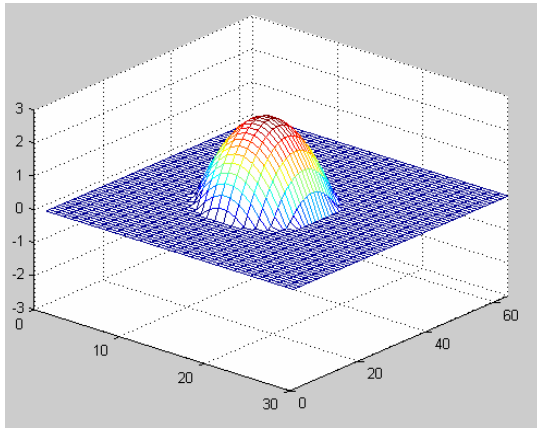


Figure 7: Epanechnikov kernel used to weigh the object's histograms – in order to include only pixels around the centre, the actual kernel bandwidth is reduced by 30% of the original size from each side

2. *Computing log-likelihood ratio:* considering two classes, namely object and background, a log-likelihood ratio of their distributions is computed according to (13); the number of bins is chosen to be 32, which is the maximum value for i in the equation (see Figures 3 through 9 in the Appendix – as a result of the non-linear function in (13), the graphs on the right of each figure show positive values for bins more likely to belong to the object, and negative ones for those likely to correspond to the background)
3. *Back-projecting likelihood values into the image space:* based on the likelihood ratio values, brighter pixels in likelihood images correspond to the object, and darker ones to the background (Figures 10 and 11 in the Appendix - since log-likelihood is computed on image regions restricted to background, LLH images also contain only those regions)
4. *Calculating likelihood histograms:* from likelihood images, likelihood distributions for object and background are estimated through histograms; these distributions are the input to the variance ratio measure
5. *Sorting features based on AVR score:* for every feature, the augmented variance ratio of its likelihood distributions is evaluated – 3 best-ranked features are selected for tracking

Once the most discriminative (based on the AVR measure) features have been found, best 3 are used for tracking – the final offset from object's original position as found by mean-shift algorithm is averaged over those features.

6. Coordinate transformation

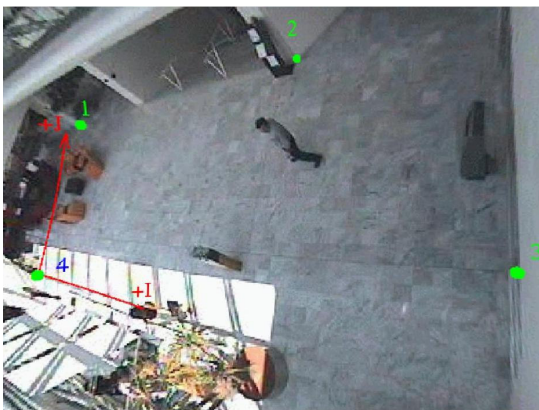
An important part of the system is the mapping of the coordinates of the objects in the original video stream to coordinates in a modelled surface of the room. This modelled surface is a top-view image of the room and can be created using standard transformation methods. The given sample video's are all shot with the same fixed camera, with an attached fish-eye lens. The advantage of such a lens is the wide view angle resulting in an almost full view of the room. The disadvantage of using such a lens is the distortion that will occur in the corners of the resulting images. We also obtained world coordinates for four points in the video frames. A frame with these points present is depicted in figure 7.a.

Given the four world coordinates (which are all located on the floor) we are able to do a projective transformation to obtain a top view of the room. But before this can be done accurately, the distortion of the fish-eye lens need to be removed first. The type of distortion that occurs when using a fish-eye lens is radial. To remove the distortion, we used the *imlenstransform()* function provided by [23]. The relevant parameters required for this function are the original image, the position in the image that represents the lens centre, and the radial distortion parameter. Since the centre of the image corresponds with the lens centre, the only parameter that is unknown initially is the radial distortion parameter. We experimented with different values for this parameter until the transformation looked correct. (check the *imlenstransform* function for more information about the parameters). We considered the transformation as a success when all the lines that are straight in real life, are also straight in the picture. In figure 7.b the final undistorted image is shown. From this picture we can see that curved lines in the distorted image became straight lines. Note that due to the transformation a small part of the border of the original image is cut of. This is the case because some points of the original image will transform to a location outside the image canvas. Because we wanted to keep the size of the original and resulting image constant we cropped the

resulting image so that the image sizes were equal again.



a



b

Figure 7:

a. Frame of the sample videos with fixed fish-eye lens. The world coordinates of the points 1,2,3 and 4 were given

b. The undistorted version of the frame depicted in figure 7.a. The unwanted effects of the fish-eye lens are removed

After removal of the lens distortion we applied a projective transformation. The standard Matlab functions `imtransform` and `cp2tform` were used to accomplish this. First, `cp2tform` was called with these two parameters: (1) The `input_points`, which are the 2D image pixel coordinates of the four given points, (2) The `base_points`, which corresponds to the given 3D world coordinates. (In fact these are also 2D coordinates since all the points lie on the floor and therefore the height is fixed). The output of `cp2tform` is a structure which contains a transformation matrix of size 3x3 which can do the transformation for each coordinate (3x1) in the lens-corrected-image (figure 7.b) to the top-view projected image. To actually transform the image in

figure 7.b, the function `imtransform` is called which expects these parameters: (1) the image to be transformed (in our case the image in figure 7.b), (2) the structure containing the transformation matrix, generated by the `cp2tform` function. In figure 8 the result of the projection is depicted.



Figure 8: Result of the projective transformation (top view) of the image in figure 7.b

Given the world coordinates (`world_p`), image coordinates (`image_p`) and the image to be transformed (`image`) the next two lines of Matlab code performs the desired transformation:

```
TFORM = cp2tform(image_p, world_p,
    'projective');
newIm = imtransform(image, TFORM);
```

The image in figure 8 is used as a container for plotting the trajectory. At each processed frame, the tracker returns the coordinates of the object that it tracks. Tracking is done in the original “fish-eye” space, so every coordinate that the tracker returns need to be transformed in two steps. In the first step the coordinate is transformed from the distorted to the undistorted space (from the space in figure 7.a to the space in figure 7.b) using the transformation functions in `imlenstransform`. In the second step the transformation matrix generated by `cp2tform` is used to transform the coordinate to the top-view space (from the space in figure 7.b to the space in figure 8).

Note that for the entire tracking process the top view image needs to be constructed only once using a “background frame” of the video. For this we used a frame that only contains the empty room. In this way the trajectories can be plotted without interference of objects in the image. To evaluate the point mapping we created a test application where the user can select points in the distorted space. The application will show where the selected point will

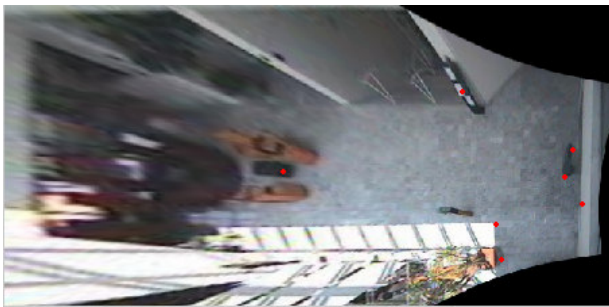
lie in the undistorted and top-view space. In figure 9 the point mapping in all three different spaces (distorted (a), undistorted (b) and top view (c)) is given for a number of points.



a



b



c

Figure 9:

- a. Distorted space, in this space the user can select points
- b. locations of transformed coordinates in the undistorted space
- c. Top-view space. The points present in this image are the transformed points from the undistorted space (figure 9.b)

The first time the transformation component is started it takes a long time to finally construct the flat top-view image (To construct Figure 9.b and 9.c for an input image of size 384x288 it takes approximately 50 seconds). As said before the actual transformation of all image points is only needed one time for a full tracking process. So it is not considered a problem. At initialization time of the tracker the construction of the top-view image is performed. After this, only the transformation of single points is needed since we only get one coordinate from the tracker at a time (the coordinate of the object that it tracks). Since the transformation matrix and the mapping from distorted to undistorted space is already calculated this operation can be applied very quickly.

7. Background subtraction

One large part of the surveillance system is the background subtraction and object detection part. Making use of background subtraction, we try to identify all objects in an area to be able to track them. This means recognising objects apart from the background and keeping track of their locations.

There are a few problems we have to take care of for this part. First we need a way to identify objects and their locations. Sometimes this can be quite hard, because of noise in the image and people standing close together. In the first case there is the risk of pixels which are just noise to be recognised as objects. In the second case we will probably identify two objects as one.

When this background is subtracted from the image (figure 10.b), the result is a grey-scaled image with a black background and all foreground objects in grey (figure 10.c). This image contains a lot of noise because of difference in illumination or little differences between the background in the background and foreground image. To get a better segmentation between the foreground and the background image, the greyscale image is compared to a threshold value ϵ which results in a black and white image. When ϵ is chosen correctly you can get rid of a lot of noise in this way (figure 10.c). For this project $\epsilon = 0.2$ was used. This means that all pixels with grey values $G > 0.2, G \in [0,1]$ are set as white. All other values will be black. This results in figure 10.d.



a



b



c



d

Figure 10:

- a.** Background used for background subtraction
- b.** Video frame from which to create separated foreground
- c.** Subtracted image in grey scale
- d.** Mask from subtracted image ($G > 0.2$)

The second part we have to take care of is the detection of non moving objects. The general idea of the system will be that it can identify objects being left behind or people showing odd behaviour. For this we need to find a way to identify non moving objects. When detected odd behaviour or suspicious objects turn out to be nothing of alarm however, we will need them to be merged into the background used for the subtraction so the object won't be recognized next time.

Background subtraction will be done in a few steps. The first one is to create a generic background. This is done by getting a number of empty or almost empty frames of the background and averaging over them. This gives us a background as shown in figure 10.a.

This resulting mask still contains some noise. Because we want exact segmentation of foreground objects we want to get rid of all noise in the mask. As can be seen in figure 10.d most of the noise only exists of lose white pixels on the black foreground, while the real foreground objects exist of larger groups of white pixels. With this information we can use two standard image processing techniques to get rid of the noise. These techniques are erosion and dilation.

Erosion is used to remove all small noise pixels in the mask. It works on the edges of the white image area's and shrinks the white area's by sort of "stamping" a black shape (called the structuring element) on the edges. When the structuring element has the right size and shape, all noise will be removed, without removing the real objects. For this project a disc formed structuring element with radius one is used for erosion. When this operation is applied to the mask of figure 10.d the result of figure 11.a is gained.

Because of the erosion, some objects are split up into separate parts. Because we want each object to be displayed as one white area, we need to dilate the image. Dilation is just the inverse operation of erosion. You take the edges of the white areas and "stamp" a structuring element over these edges to enlarge the white area. To be sure each object is displayed as one, a disk formed structuring element of radius twenty is used for this operation. This results in figure 11.b.



a



b

Figure 11:

- a. Eroded mask image
- b. Dilated image

We now use this image to locate all objects. To do this, the Matlab function *bwlabel()* was used. This function searches for all connected white area's in a black/white image and gives the locations of the centres, the sizes of the area's and the location and size of the bounding box surrounding each object. For figure 11.b this will result in two objects. In the original image however, you can see that there are actually four objects in the area. Some of these objects have merged because of the dilation operation. Because we want detection of each separate object we need to find a way to detect merged objects. This is done by using the bounding box of each object. When a new frame contains less objects than the previous frame, all centres in the previous frame are compared with the bounding boxes of the objects in the current frame. When multiple centres fall into one bounding box, you can conclude for sure that this are two objects which have merged. We now replace the centre of this the

object in the new frame with the centres of the objects in the previous frame. This way we can keep track of all separate objects, even while they are too close to each other to be tracked. The only drawback of this method is that the centres of the objects don't move while the objects are merged. This can result in wrong detection of non-moving objects. Some improvements for this problem are suggested in the future research chapter.

The next part of the system is the detection of non-moving objects. To be able to do this, a history of object locations is needed. The time an object needs to stand still before it is detected as such depends on the size of the history. The general idea is to make a vector of object locations and initialize this vector with coordinates somewhere out of the image. For each new frame, the first index of the history is discarded and the new coordinates are added at the last index. Then the locations of the first and the last history index are compared to see if there are two locations that differ less than a certain threshold. For this project we used a threshold value of forty pixels and a history of thirty frames. This means an object is detected as non-moving when it has moved less than forty pixels during thirty frames.

The greatest difficulty is the changing number of objects in the frames. Persons move in and out of the frames which means the generation and deletion of objects all the time. Because of this, the number of centres to be saved into the history keeps changing. Since a two dimensional array needs all indexes to be of the same size this gives some trouble processing. To solve this problem the number of objects of the current and the previous frame are compared. When the new frame contains less centres than the previous one, a comparison is made to see which object has disappeared. The object in the previous frame that has no centre similar to one of the centres in the new frame is said to be the object that has disappeared. This process has to be done iteratively throughout the whole history, because the names of the objects are different for different frames.

When a new frame contains more objects than the previous frames, as many new centres have to be added to the previous frames as are needed to get the same number of centres for all frames. The centres added to the history are initialised at the same location as the initial values of the history, somewhere outside the image.

```

if numberOfPrevCentres) > numberOfNewCentres
  for historySize
    centresToKeep = compare(prevCentres,
                           newCentres)
    add centresToKeep to newHistory
    prevCentres = newCentres
    get new prevCentres
  end for
end if

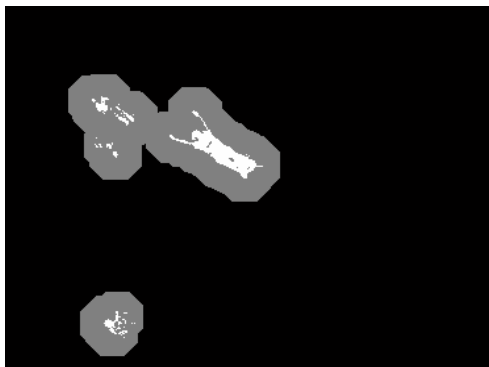
```

Figure 12: code for updating the object history

To display the separate objects nicely and possibly use them for other purposes, the “blobs” from figure 11.b should be used to create an image similar to figure 10.d. This can be done by point wise multiplying figure 11.b with figure 10.d. This operation results in figure 13.a. This image is quite similar to figure 10.d, but has all noise removed. To give a nice overview of what’s happening in the subtractor, figure 13.b can be produced by adding up figure 11.b and figure 13.a.



a



b

Figure 13:

- a.** Result of dilated image multiplied by the original mask
- b.** Addition of the dilated image and the noise-free mask

Not all non-moving objects should be detected as being hazardous. When for example a janitor puts down his cleaning cart somewhere in the area. When the object is detected as being unimportant, we want the warning to stop. To do this, the cart should be added to the background to stop detecting it. This is done by creating a new background for each iteration of the system. For this background creation, first the current frame is taken. On this frame, all objects that are not standing still are overwritten by the same location of the original background. This way all moving persons and objects are replaced by the original background. The only object not removed is the object marked as being unimportant. When background subtraction is done using this new background, the object is not detected any longer. When the object starts to move again it will be detected as a moving object and it will be removed from the background.

Non moving objects should only be alarming if they are left behind by people or suddenly appearing in the area. To be sure a person just standing still in the room for a moment isn’t detected as a suspicious object some feedback with the tracker is needed. This part is handled in the integration chapter.

8. Integration

All separate parts of the system are integrated as one whole by using a main loop. This loop loads the new frame and calls all separate methods. The first method is the background subtractor. This part returns the mask from figure 13.a and all locations of non-moving objects. The next part is the tracker. This part returns the location of the tracked object.

Because we only want to sound the alarm when someone leaves something behind (a person leaves a bag for example) we need to check if the object detected is really something new. To do this, the location of the newly detected object is compared to the locations of the objects being tracked. When the locations correspond, the non-moving object probably is a person standing still for a moment. In this case nothing needs to be done. When the object detected is not being tracked however, it probably is something left behind and there should be some reaction. In our system, a red asterisk is plotted at the location where the new object was found.

The other purpose of the locations put out by the tracker is to plot the trajectory of the person tracked on the floor map. For this, the floor map created by transforming and projecting the background image is

used. All trajectory points are converted to floor map locations and a line is plotted between those points. This way a nice path is drawn representing the trajectory of the person. The complete image also can be used to deduct who has left something behind at what time.

9. Experiments

For the tests we created an overview which contains the original image, the image with the tracker selection, the background subtracted image and the area map with the trajectory and left behind objects plotted. The overview is shown in figure 14.

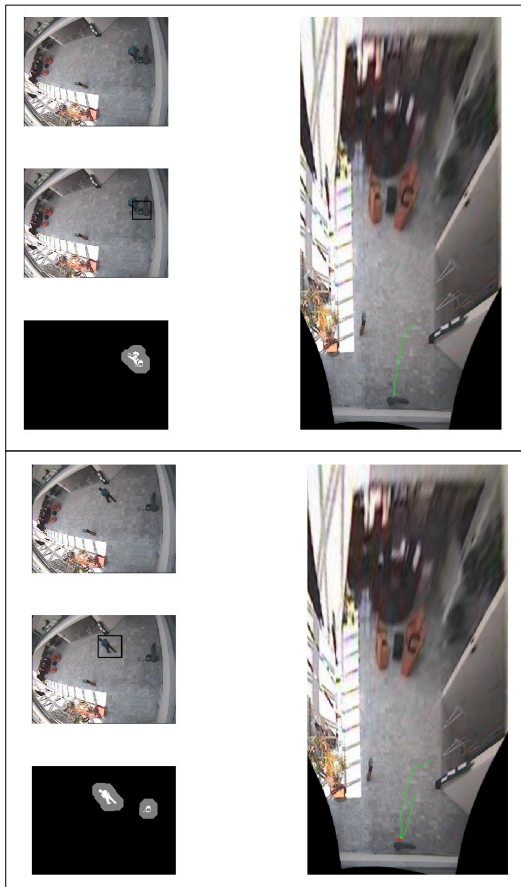


Figure 14: Samples of final system. The first image shows the path of someone walking to the dustbin. The second image shows the person walking away while he has left a bag (red asterisk).

During the testing there seemed to be some problems with the tracker. The tracker kept losing the object moving the selection down until the object was out of the selection. The tracker is very sensitive to the size of the selection window. The amount of background taken in the foreground window or foreground appearing in the background selection

confuses the tracker which makes it almost impossible to keep track.

By changing the image on which the tracker operates by making a clear segmentation between foreground and background, the results of the tracker greatly increase. The general idea is to mask the image with the background subtracted image. In this way you get a black image with all foreground objects in full colour. Because of the segmentation created in this way, the tracker can operate much easier.

The results are moderately good. When the tracker is able to keep track of the object all time it works fine. The trajectory is plotted very nicely and any objects left behind are displayed well (see figure 14.b). When the tracker loses the object however, the system tends to crash or the tracker is left somewhere on the background and keeps tracking the floor tiles. This, of course, is something that shouldn't happen and some further research will be needed to fix this problem.

10. Conclusion and Future Work

There are a few things of interest for further research. The first problem of the current tracker is the fact that it can only keep track of one person. It should be quite easy to adapt the system so it can track multiple persons or objects. Furthermore it is quite inefficient to manually select all objects to be tracked. Manual selection in combination with multiple objects to be tracked is very slow and largely reduces the efficiency of automatic tracking. Instead of manual object selection this should be done in cooperation with the background subtractor. You could think of a system that detects all objects entering the area at the edges of the screen and use the centres of these objects to automatically create selections for the tracker. The only problem of such a system would be that objects left behind at the edges of the screen would be seen as new objects and are added to the tracker. This would seriously distort the detection of non-moving objects.

An other problem stated before is the movement of merged objects. Objects that get too close together will be seen as one object. Because of the technique used to split up merged objects by replacing the merged centre by the previous centres, the objects will be seen as standing still. When the objects don't split up again within some timeframe, the merged object will get too far away from the original locations and the system will lose the objects. This effect could be reduced by for instance adding the movement vector of the merged object to the

separate objects. This way you still get the average movement of the separate objects and probably can keep track of the merged object as being multiple separate ones much longer.

At last there could be much improvement in the speed of the system. The Matlab implementation used for this project is far too slow to create a real-time system. The complete tracking process will take somewhere around ten seconds per frame on a 3.0 GHz system. With this speed it is impossible to do real-time tracking. An implementation in C and some optimizations would probably do the trick and enable the system to run in real-time.

Except for some problems we had during implementation and testing, we can conclude that the results we achieved in such a short time span are quite satisfactory. The system has been shown to work. The main problems still occurring are because of the flawed tracker we got delivered at the start of the project. This tracker and the transformation of fisheye images to the flat ground map took relatively too much time. Because of this we didn't have enough time to implement everything we wanted to, but the final result still is quite nice.

The system is able to track an object and detect objects left behind. With some tuning, debugging and adaptations like mentioned in the future research part it should be possible to get a good working system.

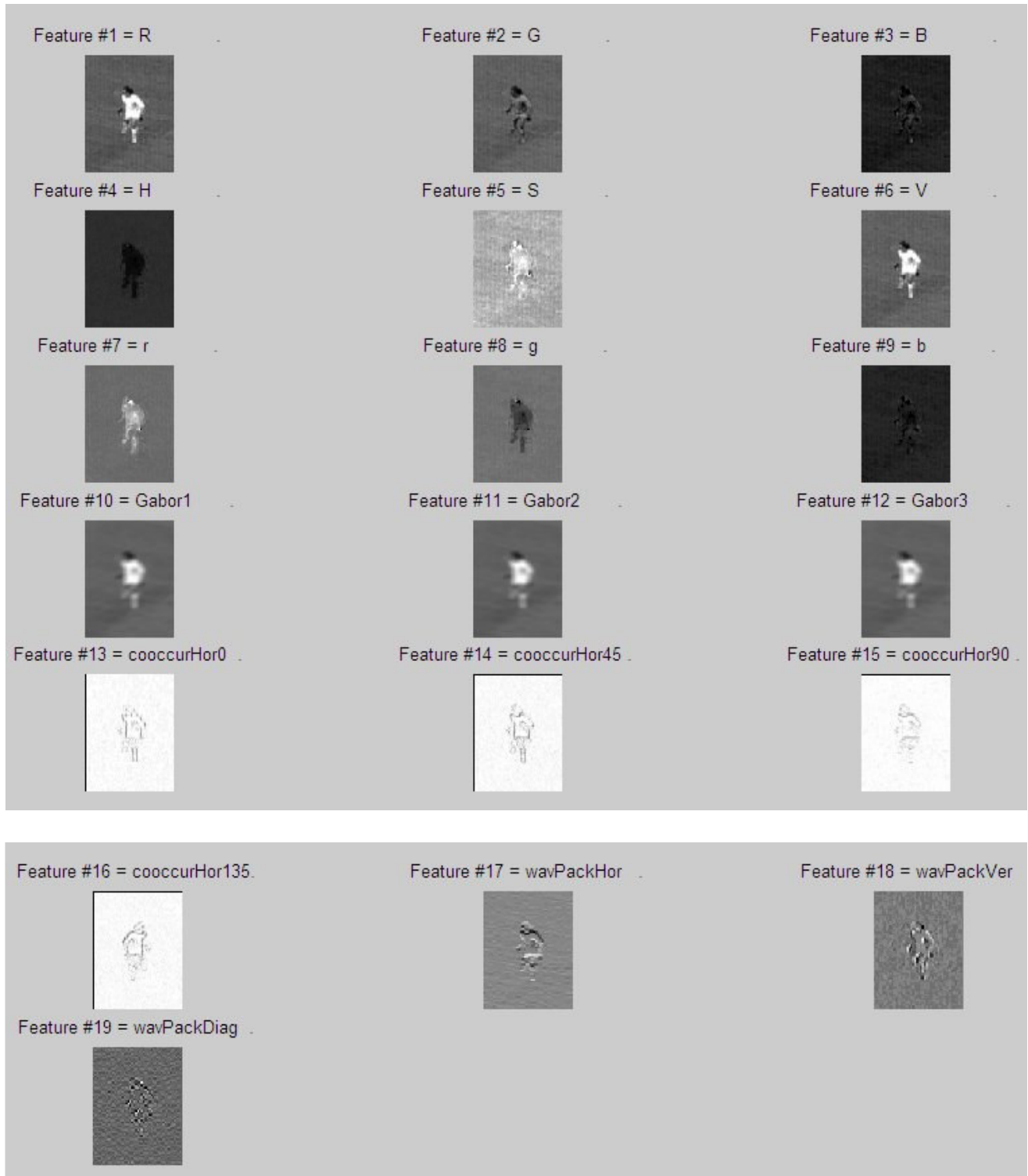
References

- [1] D. Comaniciu, V. Ramesh, P. Meer, "Kernel-Based Object Tracking", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol.25, No. 5, 2003.
- [2] D. Comaniciu, V. Ramesh, P. Meer, "Real-Time Tracking of Non-Rigid Objects using Mean Shift", *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR '00)*, vol.2, 2000.
- [3] Vladimir Nedovic, "Tracking moving video objects using mean-shift algorithm", project report, Informatics Institute, University of Amsterdam, available online at <http://student.science.uva.nl/~vnedovic/meanShift.html>
- [4] Robert T. Collins and Yanxi Liu, "On-line Selection of Discriminative Tracking Features", In *Proc. of the 9th IEEE Int'l Conf. on Computer Vision (ICCV '03)*, 2003.
- [5] Zoran Zivkovic and Ben Krose, "An EM-like algorithm for colour-histogram-based object tracking", In *Proc. IEEE Conf. Computer Vision Pattern Recognition, CVPR 2004*, 2004.
- [6] Okuma et al., "Automatic Acquisition of Motion Trajectories: Tracking Hockey Players", In *Proc. of SPIE Internet Imaging V*, pg.202-213, January 2004.
- [7] Trygve Randen and John Hakon Husoy, "Filtering for Texture Classification", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 21, No.4, April 1999.
- [8] Mihcak et al., "Low-Complexity Image Denoising Based on Statistical Modeling of Wavelet Coefficients", *IEEE Signal Processing Letters*, Vol. 6, No.12, December 1999.
- [9] Stephane G. Mallat, "A Theory for Multiresolution Signal Decomposition: The Wavelet Representation", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 11, No. 7, July 1989.
- [10] R. Fisher., "CAVIAR: Context Aware Vision using Image-based Active Recognition" <http://homepages.inf.ed.ac.uk/rbf/CAVIAR/>
- [11] H.T. Nguyen, A. W.M. Smeulders, "Template tracking using colour invariant pixel features". In *Proc. of the Inter. Conf. on Image Processing, ICIP'02, Vol 1, pp. 569 - 573, Rochester, 2002*.
- [12] H.T. Nguyen, M. Worring and R. van den Boomgaard, "Occlusion robust adaptive template tracking", In *Proc. of the IEEE Inter. Conf. on Computer Vision, ICCV'01, Vol.1, Vancouver, 2001*.
- [13] David Vignon, Brian C. Lovell, Robert J. Andrews, "General Purpose Real-Time Object Tracking Using Hausdorff Transforms", (2002) In *Proc. of the IPMU2002, pages 1-6, Annency, France*.
- [14] D.M. Gavrilu, "Multi-feature Hierarchical Template Matching Using Distance Transforms", (1998) In *Proc. IEEE International Conference on Pattern Recognition, Brisbane, Australia*
- [15] Steven Mills, Tony Pridmore and Mark Hills, "Tracking in a Hough Space with the Extended Kalman Filter", (2003) In *Proceedings of the International Conference on Pattern Recognition (ICPR2004), August 2004*
- [16] Dorin Comaniciu and Visvanathan Ramesh, "Mean Shift and Optimal Prediction for efficient object tracking", *IEEE Int. Conf. Image Processing (ICIP'00), Vancouver, Canada, Vol. 3, 70-73, 2000*
- [17] M.Brown, D.G. Lowe, "Recognising Panoramas", *Tenth International Conference on Computer Vision (ICCV 2003), Nice, France, (October 2003)*
- [18] R. Szeliski, H. Shum, "Creating Full View Panoramic Image Mosaics and Environment Maps", In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*

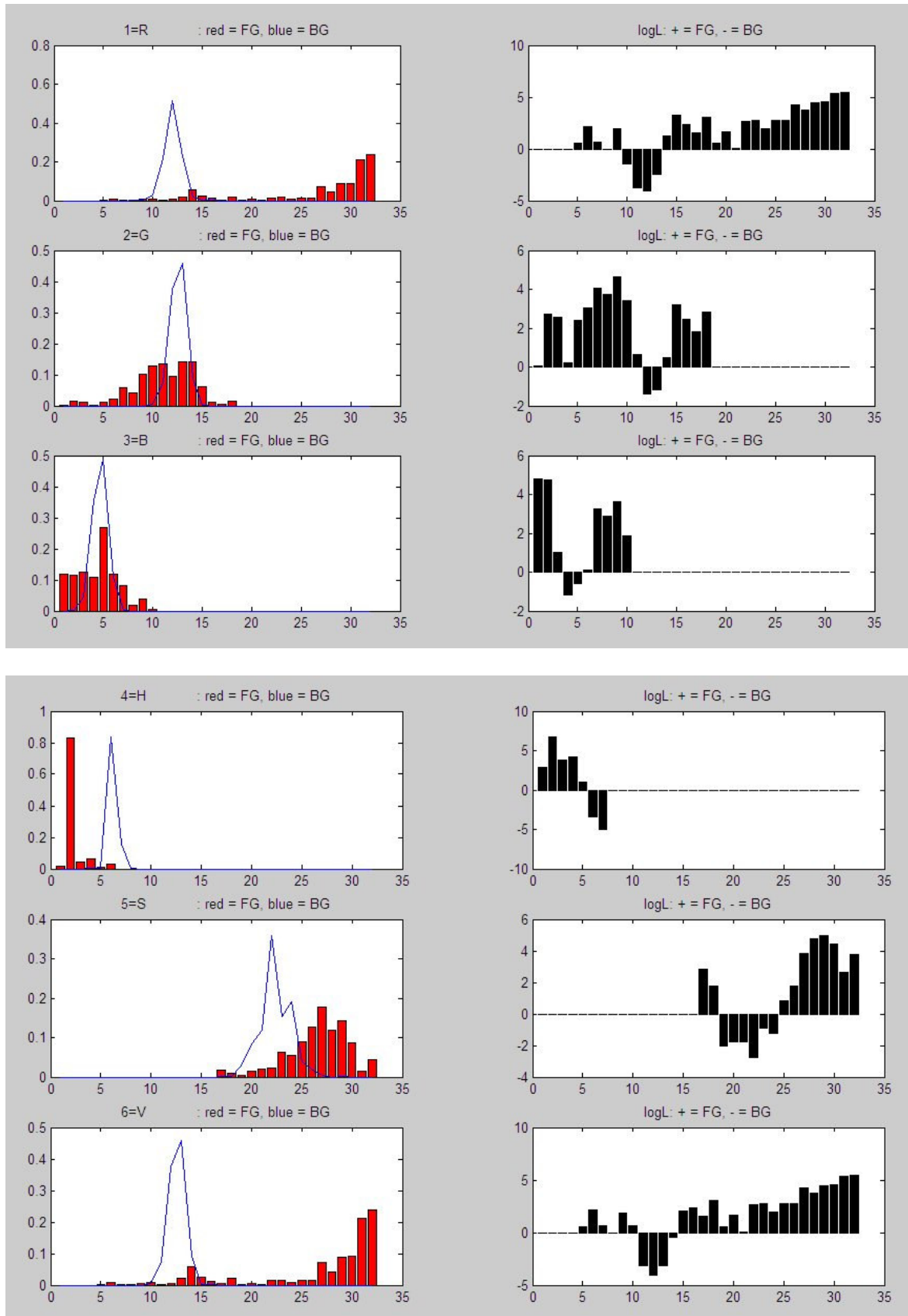
- [19] LIU Ya, AI Haizhou, XU Guangyou, "Moving Object Detection and Tracking Based on Background Subtraction", (2001) *In Proceedings of SPIE Vol. #4554 [4554- 11], October 2001*
- [20] T. Gevers, "Color in Image Search Engines", In *Principles of Visual Information Retrieval, Springer-Verlag, London, February 2001.*
- [21] Trygve Randen, Håkon Husøy, "Filtering for Texture Classification: A Comparative Study", In *IEEE transactions on Pattern Analysis and Machine Intelligence, Vol. 21, No. 4, April 1999.*
- [22] Ahmad Poursaberi, "Matlab 2D Gabor filter", <http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=5237>
- [23] Peter Kovsesi, "Matlab imlenstransform function", University of Western Australia, <http://www.csse.uwa.edu.au/~pk>

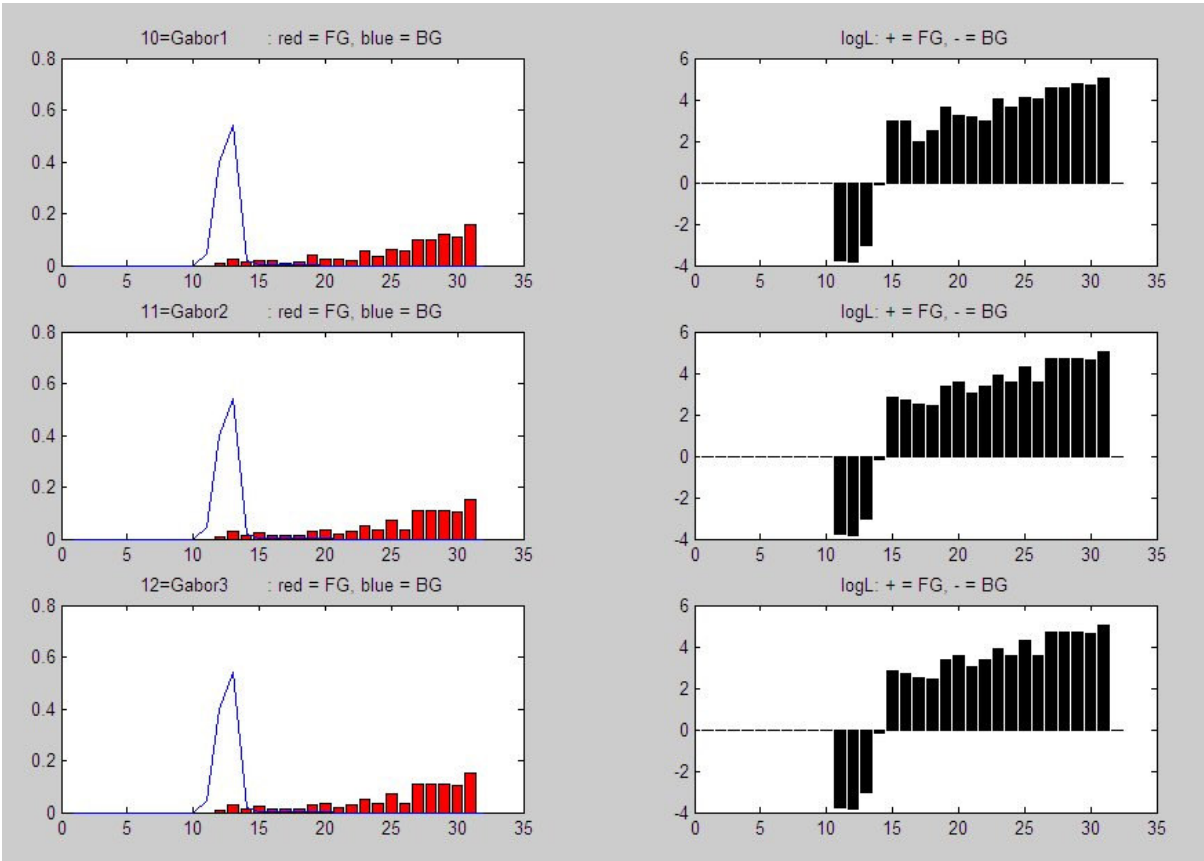
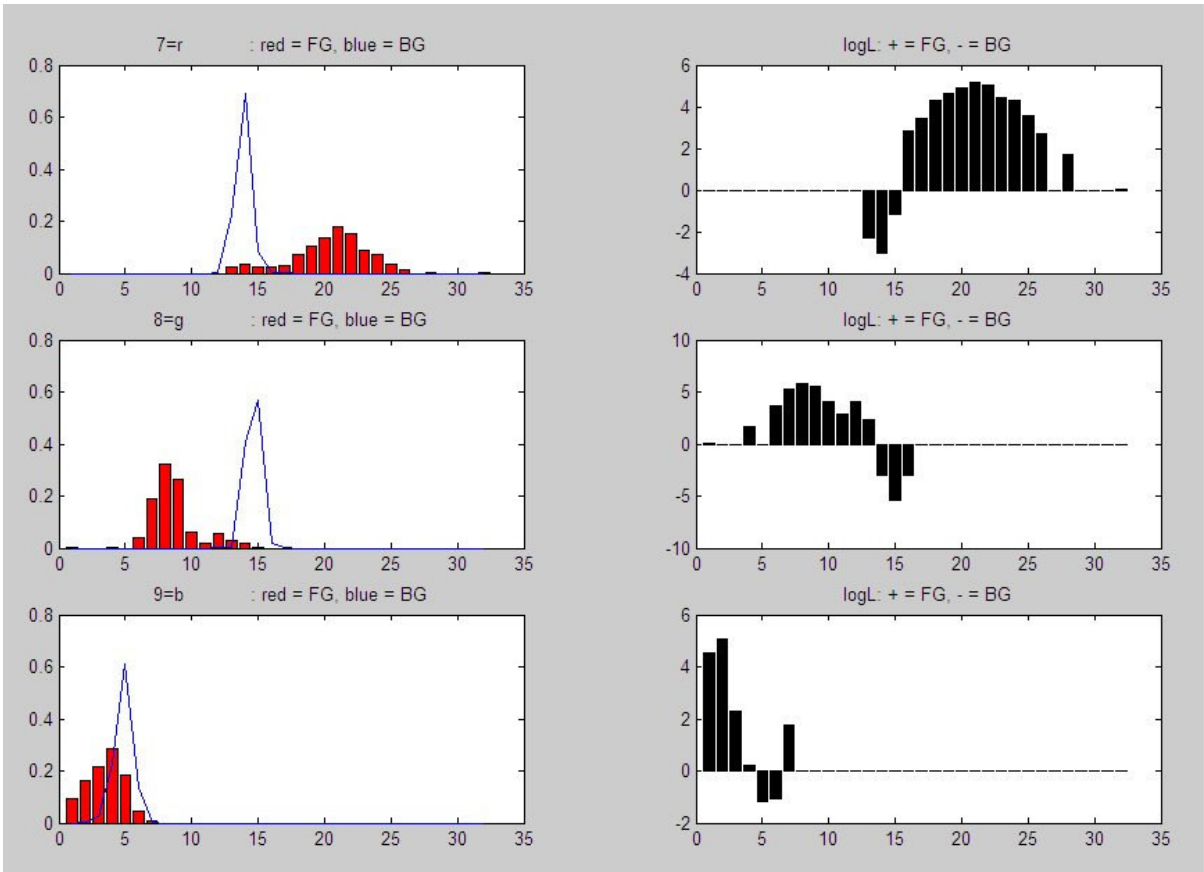
Appendix – Figures resulting from various steps of adaptive feature selection process

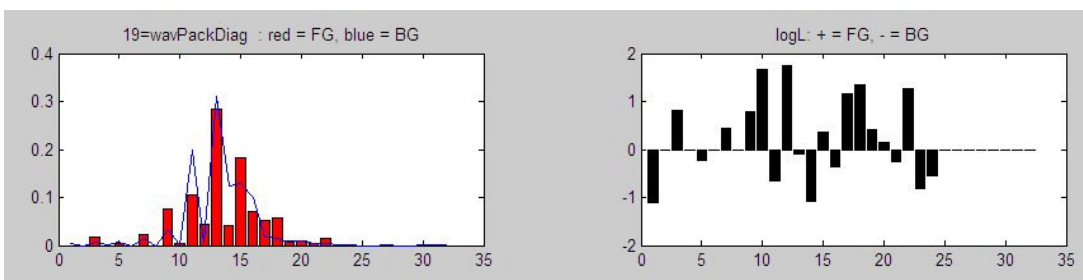
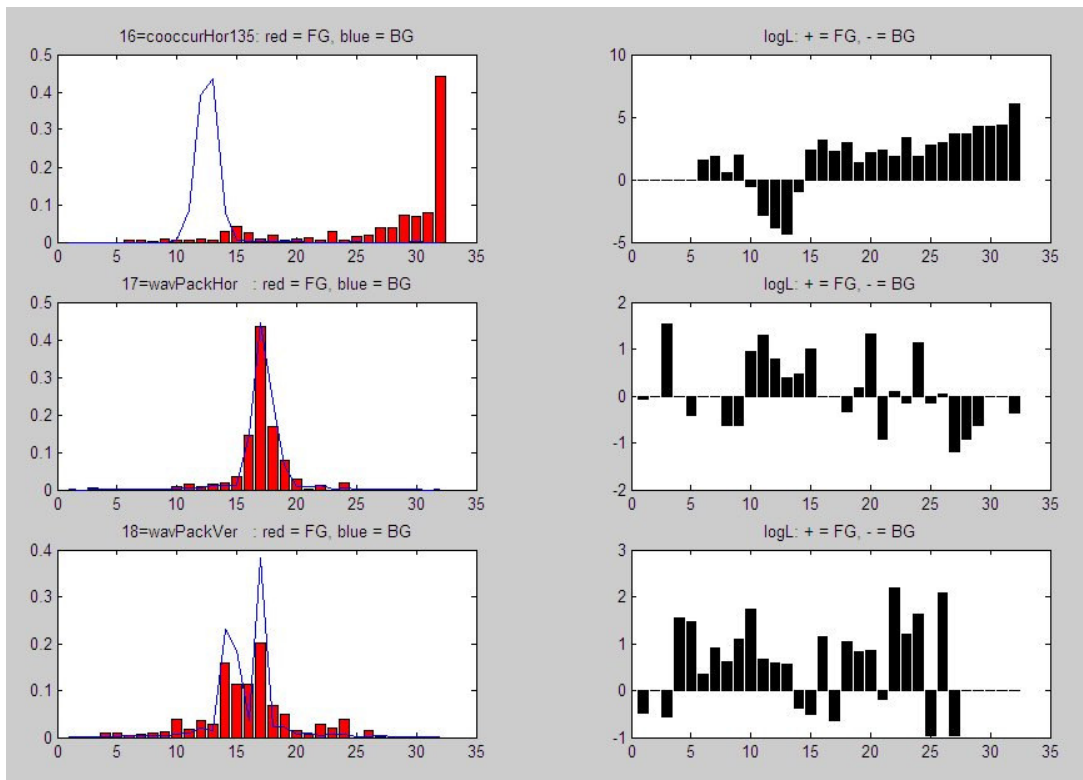
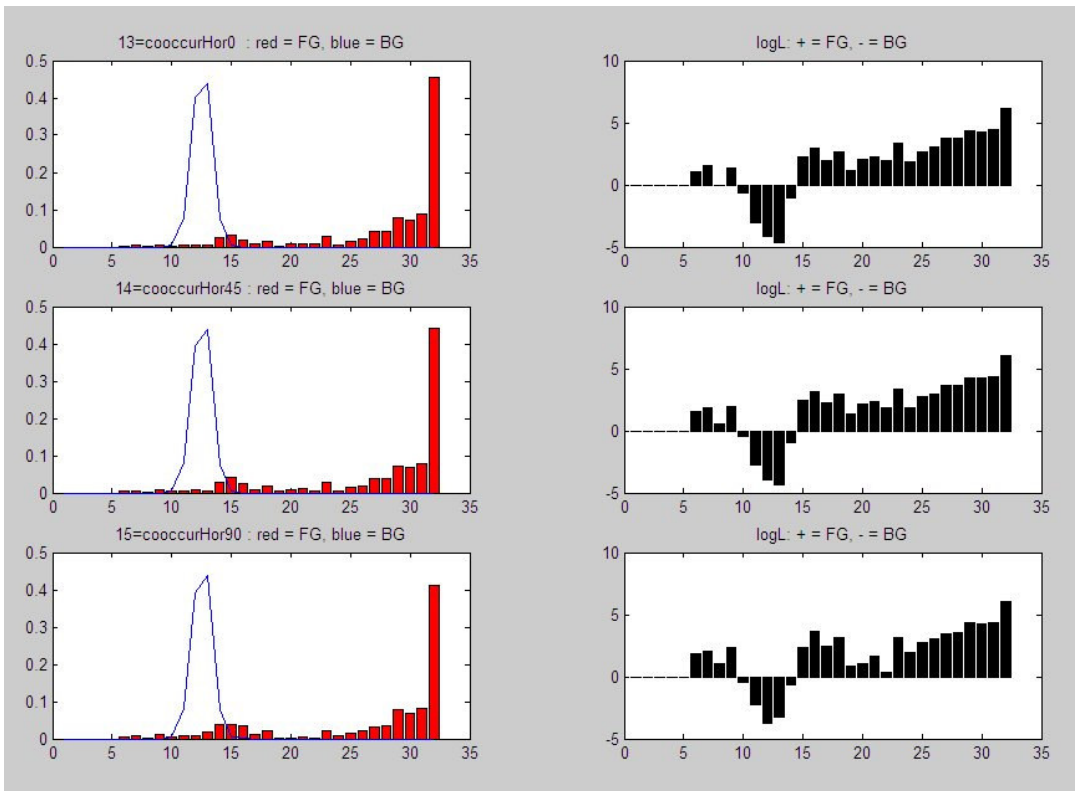
Figures 1 and 2: Extracted features – RGB, rgb, and HSV colour components, three Gabor filtered images, four co-occurrence images and three wavelet-packet images



Figures 3 - 9: Distributions of original features represented by histograms – red bins correspond to the object and blue graph to the background; on the right is the likelihood ratio for those features – positive values indicate object’s and negative ones background’s likelihood







Figures 10 and 11: Likelihood values, back-projected into the image space, ranked based on the AVR score of their likelihood distributions

