University of Edinburgh
School of Informatics

Combining Multiple Detection and Tracking Methods

4th Year Project Report
Computer Science

Jonathan Miles (0451501)

April 1, 2009

Abstract: Tracking moving targets in video image data is a common application of computer vision. In situations where the video camera position is stationary, moving objects can be distinguished by comparing new frames with a representation of the background scene. This representation is referred to as the *background model*. Various target detection techniques exist which uniquely model the background. In general, existing work uses only a single detection technique to identify target locations. Four unique target detection algorithms are implemented. A system is developed which allows the stages of target detection and tracking to be executed in a modular manner. The system accepts the data from the four techniques as input, and utilises five strategies of data combination. The results are evaluated, to investigate which strategy provides the most accurate performance, and to ascertain whether multiple combination technique provide more accurate results than the component methods. Results are obtained for five combination strategies, and four individual detection methods. It is seen that the combination strategies evaluate as more accurate than any of the component methods.

Acknowledgements

Many thanks to Professor Bob Fisher, my supervisor for this project, whose support and feedback were invaluable throughout. Thanks also to the creators of the software used in this project: The MathWorks, for MATLAB, and also the Python Software Foundation. Additional thanks to all the simple carbohydrates and Taurine-based beverage consumed during the report-writing process: the sugar-rush was appreciated, the heart palpitations were not.

Contents

1. Introduction

Tracking moving targets in video image data is a common application of computer vision. In situations where the video camera position is stationary such as in surveillance footage, moving objects can be distinguished by comparing new frames with a representation of the background scene. This representation is referred to as the *background model*.

Various methods may be used to produce the background model, and the method and features used will affect the accuracy of target detection. Producing a successful background model requires adapting to or tolerating changes to the scene and analysing whether these changes represent moving objects worthy of detection and classification. Changes to the background may happen at a small (*local*) scale or at a large scale encompassing the whole scene (*global*). These changes may be gradual, such as the slow change in lighting caused by the movement of the sun in an outdoor scene, or immediate, such as the switching off of a light in an indoor scene. There may be local motion caused by the movement of objects such as branches or blades of grass swaying in a breeze, and there may be global motion as the camera is shaken by the wind or a passing vehicle.

In addition to these problems, difficulties may be caused by shadows moving due to changes in lighting direction or foreground object position, or by objects being removed from or added to view, and becoming part of the background after being stationary for a extended period of time. Many differing Background Modelling Techniques have been proposed that attempt to surpass these problems and adapt to these changes, which rely on modelling features such as pixel intensity (brightness), edges, or block correlation.

Each of these approaches varies in how well they deal with the previously mentioned changes and problems in the scene, and work in this field typically suggests extensions of existing techniques, or the development of new background modelling systems. This project aims to contribute to the field by exploring the potential of combining multiple existing background modelling techniques to produce a system which detects targets with greater accuracy than any of it's component systems.

There is the potential to combine these individual approaches in many ways. In this project, we investigate several strategies and evaluate their success.

1.1 Project overview

The primary goal of the project was to construct a target detection system capable of supporting multiple foreground detection methods, and with the ability to combine their output in multiple ways, and at multiple stages of the target detection and tracking process.

The system structure developed was highly modular, and allowed combination methods to combine data at any level of the system hierarchy. The component algorithms developed performed functions that included *Foreground Detection*, *Foreground Cleaning*, *Bounding Box Extraction*, *Tracking of Targets* and *Merging of Tracks*. Five routes in the system were constructed, each with a unique combination strategy, and each taking four background modelling techniques as input, namely: *Static Background Subtraction*, *Frame Differencing*, the *Mixture of Gaussians* method, and *Nonparametric Kernel Density Estimation*.

The performance of the system was evaluated by inputing multiple image sequences, and comparing the data output to a ground truth data set for the appropriate test case scenario. Measures of accuracy were decided upon, and a series of tests was run, producing an extensive results set.

In the course of this project the goal is to gather evidence to either support or refute the hypothesis that using combination strategies (as previously described) will give greater performance than any of the individual component techniques. Improved performance is expected to be possible but the degree of improvement cannot be ascertained until the results of combination experiments are collected and evaluated.

1.2 Hypotheses of the project

This section explicitly states the claims of the project and evidence needed to support them. The goal of this project is to investigate the performance of a system employing a combination of multiple background modelling and target detection techniques.

1.2.1 The primary hypothesis

Combining detection methods gives more accurate results than the component methods do individually.

This hypothesis may be tested by producing results for the combination methods and for the individual detection methods, for a wide range of parameters.

1.2.2 Main conclusion summarised

The conclusion gathered from the results of the system built is that the combination produces good results, but definite superiority of the combination strategies over the individual methods is called into question.

1.3 Structure of this document

To provide background on the project, we first present a literature review of related research papers, found in Chapter 2. This related research largely consists of a number of background modelling and target detection methods. The third Chapter describes the image data used in the project, comprising of the CAVIAR and BEHAVE data sets. Chapter 4 provides an overview of the implemented system, describing it's structure and the data combination strategies it makes possible, and Chapter 5 describes the algorithms developed for the system in detail.

Chapter 6 then details the methodology employed in testing the performance of the system, and the implementation of the algorithms neccessary to do so. The results produced by this performance testing methodology are given in Chapter 7, which also contains analysis of this data. The final chapter discusses the work undertaken, suggesting conclusions which may be drawn, and further work which may be carried out in the future.

2. Related Research

This Chapter details relevent work in the area of the project. The majority of work in the field typically suggests extensions of existing background modelling techniques, or the development of new background modelling systems.

2.1 Background detection methods

The related work in this area centres on developing models of the background of a scene from image data. The most basic possible method that would achieve this is to choose a frame of the scene with no foreground objects contained in the image. New frames could be compared with this 'empty frame' by subtracting the colour values of the current image frame from the static empty background frame, and finding the absolute difference between the two for each pixel. If the difference is greater than a specified threshold value, then that pixel is judged to be in the foreground, otherwise it is judged to be in the background. Another simple background modelling method uses the previous frame, or the nth previous frame as the background image, and performs the same thresholding technique as previously mentioned. This crudely accounts for changes to the background model.

A slightly less basic technique that accounts for changes in the background is the N-frame median background model. This dynamically records the previous n frames, and then the background model is generated from the median colour value for each pixel in these n frames. Therefore the colour for each pixel is the most common colour seen in that location.

A more advanced technique called the Basic Gaussian Model also keeps track of a certain number of frames, and the pixel intensity values for each frame. This model is used in *"Pfinder: Real-time tracking of human body"* by *C.R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland*[9], and assumes 'a relatively static situation such as an office, and a single moving person '. In this model, noise, or background movement in the image is represented by a zero mean Gaussian distribution with a full covariance matrix, where colour values are recorded as Y,U,V values. The statistics for the average value for each pixel a recursively updated using a simple adaptive filter, which allows for gradual changes in lighting and (the authors claim) movement of objects in the scene.

An extension of this method was proposed by *W. E. L. Grimson, C. Stauffer, and R. Romano* in *"Using adaptive tracking to classify and monitor activities in a site"*[3]. They assert that a pixel is best modeled by a mixture of n Gaussians, rather than the single Gaussian distribution suggested by the Pfinder paper. They state that multiple processes are likely to be observed at a particular point in a scene over time, and use the examples of an area of swaying tree branches, where sometimes the pixel will observe the branch, and sometimes it will observe the scene behind the branch. Alternatively, an area of rippling water may sometimes reflect the sky, sometimes reflect the horizon, and sometimes appear the colour of the water. This approach has also been utilised in *"Image segmentation in video sequences: a probabilistic approach"* by *N. Friedman and S. Russell*[2]. In this paper, the Mixture of Gaussians method is used to model pixels in a scene showing cars on a motorway, and a supervised technique they describe as an 'efficient incremental version of EM (Expectation Maximum Algorithm) finds estimates for the maximum likelihood of parameters in probabilistic models.

In situations such as the camera watching a motorway, the pixels can be realistically determined as being in only several classes, such as 'Car', 'Shadow' and 'Background'. In *"A probabilistic background model for tracking,"* by *J. Rittscher, J. Kato, S. Joga, and A. Blake*[7], their representation is based on a Hidden Markov Model, where the hidden states of the model denote foreground, background and shadow. In this model however, the states have to be included manually. The authors claim a unique benefit of Hidden Markov Models is that a temporal continuity constraint is imposed. This effectively means that is a pixel is identified as being in the foreground, it is expected to stay as part of the foreground for a certain period of time before it reverts to the background. Another related technique, known as the Hidden Markov Model Using Multiple Global States can be seen in *"Topology free hidden markov models: Application to background modeling"* by *B. Stenger, V. Ramesh, N. Paragios, F. Coetzee, and J. Bouhman*[8].

Another method that is often used is the detection of edge features. This is an interesting technique in that illumination changes in the scene don't affect the outcome of the technique to any great extent. This is described in detail in *Y.-H. Yang and M. D. Levine*'s *"The background primal sketch: An approach for tracking moving objects"*[10], and also in *"Detection and location of people in video images using adaptive fusion of color and edge information"* by *S. Jabri, Z. Duric, H. Wechsler, and A. Rosenfel*[4].

Several other advanced techniques exist, such as block-based approaches, as suggested by *T. Matsuyama, T. Ohya, and H. Habe*, in *"Background subtraction* *for nonstationary scenes"*[6]. Another advanced technique which is often used is Nonparametric kernel density estimation, as described in *"Background and Foreground Modelling Using Nonparametric Kernel Density Estimation for Visual Surveillance"* by *A. Elgammal, R. Duraiswami, D. Harwood and L.S. Davis*[1].

2.2 Summary

From the previous literature review, it can be seen that a variety of well-established detection methods exist, but no existing work was found that attempted to bring together multiple background modelling techniques to produce a single output.

3. Test Case Scenarios

This Chapter describes the Test Case Scenarios used to evaluate the system. In this context, Test Case Scenarios are image sequences containing interactions of targets in a scene. In the Test Case Scenarios used, human targets are used. The Test Case Scenarios also contain a set of *Ground Truth* data, which is a set of hand-labelled data, giving locations and orientations of targets in the scene on a frame-by-frame basis. The *Ground Truth* data is seen as the 'gold standard' of results: the closer a detected target matches to the ground truth target, the more accurate a result it is judged to be. To gain a confident overall performance assessment of a detection method, it is desirable to input image sequences from multiple sources, in order to evaluate the accuracy under a variety of conditions. Differing conditions may be provided by different physical locations of the scene, differing lighting conditions, or different recording/encoding settings, for example. The following sections describe the image data used by the system.

3.1 CAVIAR

EC Funded CAVIAR project/IST 2001 37540, found at URL:

`http://homepages.inf.ed.ac.uk/rbf/CAVIAR/`.

The following information, quoted from the page above, describes the images sequences from *CAVIAR* used as input data in this project.

> The first section of video clips were filmed for the CAVIAR project with a wide angle camera lens in the entrance lobby of the INRIA Labs at Grenoble, France. The resolution is half-resolution PAL standard (384 x 288 pixels, 25 frames per second) and compressed using MPEG2. The file sizes are mostly between 6 and 12 MB, a few up to 21 MB.
>
> The second set of data also used a wide angle lens along and across the hallway in a shopping centre in Lisbon. For each sequence, there are are two time synchronised videos, one with the view across and the other along the hallway. The resolution is half-resolution PAL standard (384 x 288 pixels, 25 frames per second) and compressed using MPEG2. The MPEG file sizes are mostly between 6 and 12 MB, a few up to 21 MB.

3.2 BEHAVE

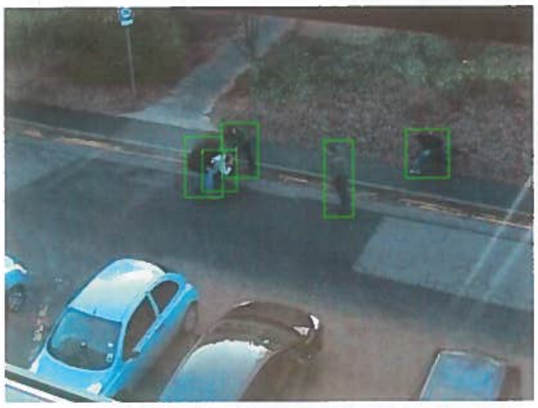
EPSRC Funded BEHAVE project GR/S98146, found at URL:

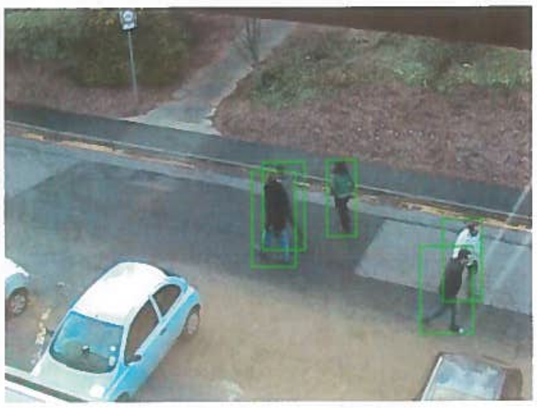
`http://groups.inf.ed.ac.uk/vision/BEHAVEDATA/INTERACTIONS/`.

The following information, quoted from the page above, describes the images sequences from *BEHAVE* used as input data in this project.

> The dataset comprises of two views of various scenario's of people acting out various interactions. The data is captured at 25 frames per second. The resolution is 640x480. The videos are available either as AVI's or as a numbered set of JPEG single image files.

In contrast to the CAVIAR data, which contains only indoor scenes, the BEHAVE image data is exclusively outdoor scenes.





4. System Overview

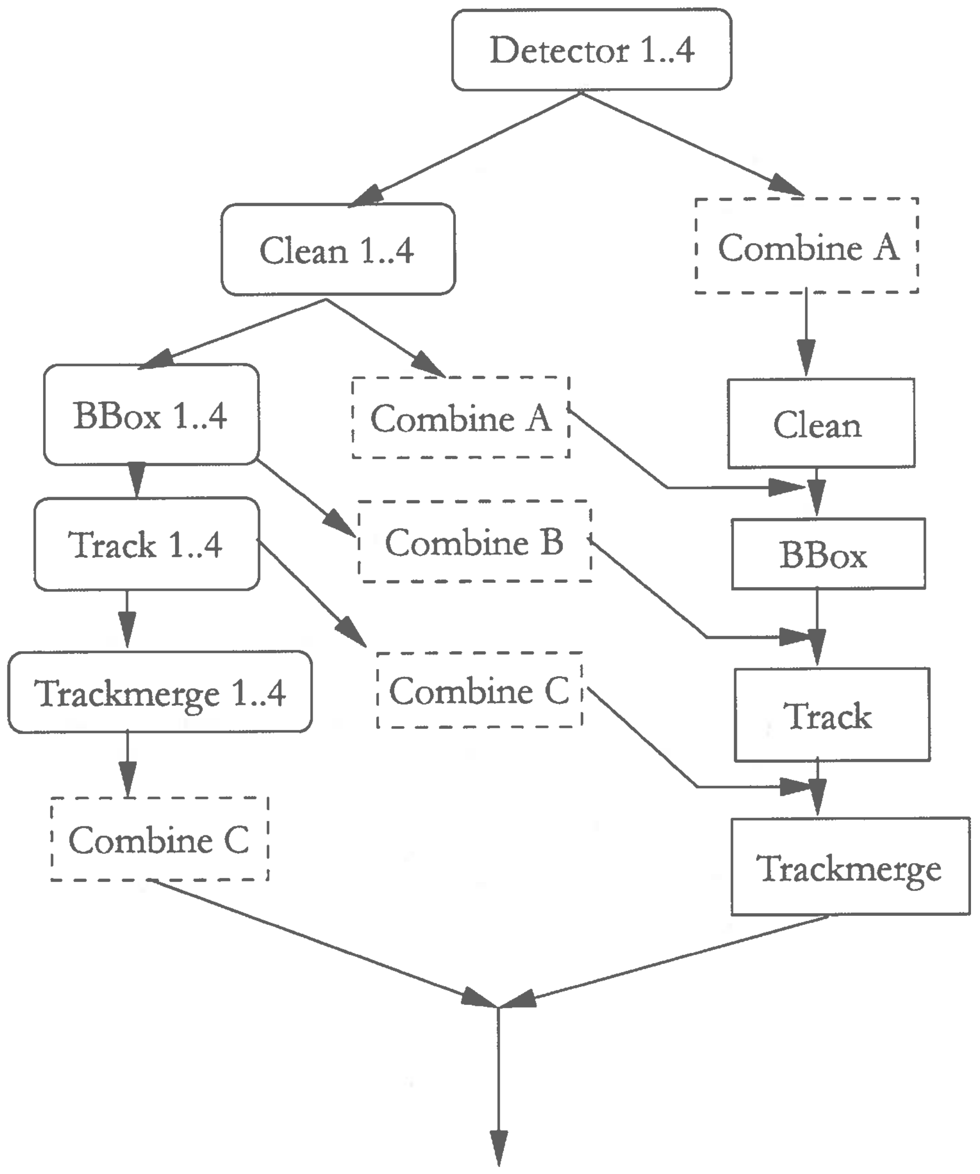
This purpose of this Chapter is to give an overview of the structure of the system, as comprised of its component algorithms. These algorithms are briefly explained (full explanations follow in the next Chapter), and the possible routes through the system structure are listed and briefly described.

4.1 Structure of the system

The system is structured in a modular nature, with the ability to individually execute the component stages of the target detection process. The identified stages are: *Foreground Detection*(`Detector`), *Foreground Cleaning*(`Clean`), *Bounding Box Extraction*(`BBox`), *Tracking of Targets*(`Track`) and *Merging of Tracks*(`Trackmerge`).

Combination of data may occur at five stages of the process. The possibility of combination occuring at five stages means that these are five possible routes through the system, which in turn means that there are five possible combination strategies. Although there are five routes through the system when combining data, there are only three *Combination Methods*(`CombineA`, `CombineB`, and `CombineC`). The reason for this is that for two of the routes, combination occurs at the foreground image stage (`CombineA`), one route combines data at the bounding box stage (`CombineB`), and the remaining two routes combine data at the track stage (`CombineC`).

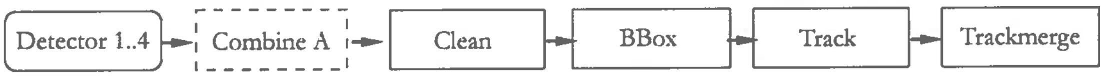
The following diagram visually depicts the structure of the implemented solution. Each box shown represents an algorithm in the system. Boxes with rounded corners, and the `1..4` suffix represent algorithms that are run four times, once on each data set produced by a detector. There were four detection algorithms implemented, so there are four data sets to operate upon. Therefore, these boxes receive multiple inputs, and produce multiple outputs. The boxes with dashed borders are exclusively *Combination Methods*, and receive multiple inputs to produce a single output. The final class of box has square corners, and this represents an algorithm that receives a single input, to produce a single output.



4.2 Possible routes

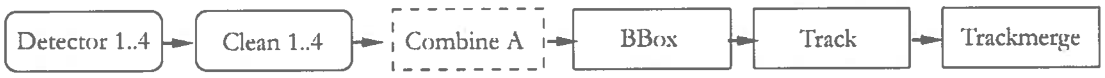
As previously explained, there are five possible routes through the system structure, which provide five strategies of data combination. Each route will be displayed visually, and a brief description will be provided.

4.2.1 Route 1



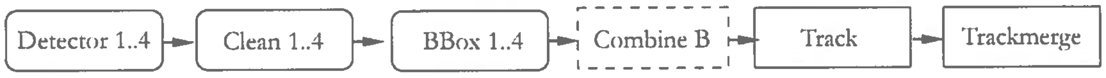
This combination strategy runs the four `Detector` algorithms, returning four binary images containing the foreground pixels from each detector. `CombineA` inputs these foreground images, and combines them to output a single foreground image. Noise in this foreground image is removed by `Clean`, and bounding boxes are then extracted from the result using the `BBox` function. Once the system has completed running these algorithms on a complete set of frames, the `Track` function goes through the bounding box data and creates tracks from overlapping frames. `Trackmerge` is finally used to link together the resulting tracks.

4.2.2 Route 2



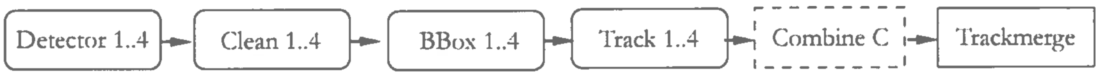
This combination strategy is similar to the last, but here, the foreground images from each detector are cleaned individually before being combined by `CombineA` to produce a single foreground image.

4.2.3 Route 3



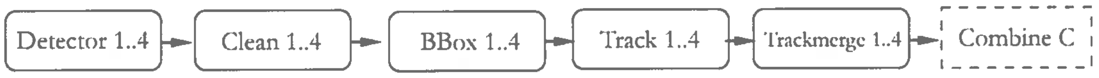
This combination strategy combines the data after extracting four sets of bounding boxes, one for each foreground image. This requires a new combination algorithm, `CombineB`, and produces a single set of bounding boxes, which are then tracked and merged.

4.2.4 Route 4



This combination strategy combines the data one stage later again, producing a set of tracks for each detection method, and then using the final combination algorithm, `CombineC`, to produce a single set of tracks which are then merged.

4.2.5 Route 5



The final combination strategy executes the entire tracking operation, up to the `Trackmerge` stage before the combination algorithm is used. As in the previous route, the `CombineC` function is used.

5. Algorithms Used in System Implementation

This Chapter describes in detail how the system and all the functions operate. All the component algorithms in the system are described, including individual *Foreground Detection* methods, *Foreground Cleaning*, *Bounding Box Extraction*, *Tracking of Targets* and *Merging of Tracks*. All functions are implemented in MATLAB[5]

5.1 Detection methods

Four *Foreground Detection* methods were chosen for combination in our system: *Static Background Subtraction*, *Frame Differencing*, the *Mixture of Gaussians* method, and *Nonparametric Kernel Density Estimation*.

5.1.1 Static background subtraction

In this method, a static representation of the background image is found by reading frames of the background scene, and finding the median colour value at each pixel location. In the implementation, colour values are represented in the *RGB* space.

Foreground regions are identified in input frames by subtracting the median background image from the new frame, and finding the absolute difference between the two. If the difference at a pixel location is above a threshold value, that pixel is identified as being foreground.

Parameters

The only variable parameter of this technique is an integer threshold value.

5.1.2 Frame differencing

This method is similar to the *Static Background Subtraction* technique, in that it also subtracts a static background image from input frames, thresholds the absolute difference, and identifies pixels as foreground if the threshold is exceeded.

The background model differs in that it does not remain static, but is dynamically chosen as the nth previous frame.

Parameters

This technique has two parameters: the integer subtraction threshold value, and an integer value which determines by how many frames the input image and the background image will be separated.

5.1.3 Mixture of gaussians

The *Mixture of Gaussians* model is initialised over a series of training frames, containing no foreground targets. These frames allow this method to store a background history for the RGB channels of every pixel in the scene. The history of color intensities is stored in a histogram, with frequency of result on the Y-axis, and channel brightness on the X-axis. A *Simple Gaussian Model* fits a single Gaussian distribution over the histogram, whereas the *Mixture of Gaussians* model fits an arbitrarily pre-defined number of gaussians over the histogram values, allowing the model to cope with multi-modal background models. These combinations of gaussian distributions give the background *Probability Density Function.* All weights are updated each frame. If the current pixel value is within a certain distance of the PDF, it is regarded as background and used to update the background model. Otherwise, it is regarded as foreground and added to the foreground representation. The gaussian classifier implemented is based upon a classifier written by *Seth Benton.*

Parameters

Parameters include the number of Gaussians to use, the initial values of a Gaussian when it is first created, and the color distance threshold value.

5.1.4 Nonparametric kernel density estimation

As with the *Mixture of Gaussians* model, the *Nonparametric kernel density estimation* model is initialised by loading in a sequence of training frames to produce a background model. The *Nonparametric kernel density estimation* method takes 50 background samples, after converting the training images into *Chromaticity coordinates.* The *Chromaticity* colour space represents colours without a luminance value, and thus is able to better adapt to targets walking into shade, for example.

Once 50 background frames have been collected, each pixel is assigned a *Sigma* value. This value is found by calculating the median change value from frame to frame. The detection method allows more variation in areas of high sigma, and this accomodates areas of the image that are more sensitive to vibration, for example.

Once the model has been initialised, detection may be performed. During detection, the background history is maintained (the method stores the last n values classified as background for a given pixel). First, the lightness of a new pixel is divided by the lightness value of every history value for that location, and the number of history values for which the result is within a set range is counted (set range may be 0.8 to 1.2 for example). If no pixels are counted then the pixel is foreground. If pixels are counted, the probability of the pixel being background is calculated.

To estimate probability of background, *Bayes Theorem* is used. We can calculate the probability of a particular colour value occuring, given that the pixel is background. We also make estimates on the overall probability of a pixel being foreground (0.99) and the probability of a particular colour being foreground (0.001). Using the *Bayes Theorem*, the probability of a pixel being background can then be calculated. If the probability is above a certain threshold (suggested value approx. 0.05) then the pixel is classified as foreground. The Nonparametric kernel density estimation detector implemented is based upon a classifier written by *Professor Bob Fisher*.

Parameters

Three parameters control the *Nonparametric kernel density estimation* model. Two of these parameters are the upper and lower bounds of the lightness ratio range (e.g. 0.8 and 1.2). The final parameter is the probability threshold on whether a pixel is classified as foreground.

5.2 Foreground cleaning techniques

The result of the detection methods is a binary image, with background pixels holding a value of '0' and foreground pixels holding a value of '1'. This image inevitably contains noise, appearing as stray pixels scattered through the image, caused by minor variations in the scene or visual noise produced by the recording device. It is desirable to remove this noise to produce a more accurate representation of the large targets in the image. The foreground cleaning techniques used in the implementation consist of a dilate operation followed by an erode operation.

The dilate operation extends the borders of foreground regions by a set number of pixels, and the erode operation does the opposite, shrinking the borders of a foreground region inwards. Used in sequence, the dilate operation closes up gaps inside regions or between very close regions, and the erode function takes the result, and shrinks the regions to their original size. A slighly larger parameter chosen for the erode function causes very small regions to be removed entirely, eliminating noise.

Parameters

The `Clean` algorithm has two parameters, one which dictates by how many pixels the foreground should be dilated, and one which dictates by how many pixels the foreground should be eroded.

5.3 Bounding box extraction

The *Bounding Box* extraction algorithm operates on a binary image representing foreground regions. The algorithm uses MATLAB library functions to isolate all the islands of foreground pixels, and extract their properties. The included property labeling function has the ability to extract statistics on the area of a foreground region, as well as other properties, such as the location of its Bounding Box coordinates. A bubble sort operation is applied, in order to sort the regions in order of descending size. These regions are then saved to an array. We iterate through this array, and while the area is larger than a supplied minimum threshold value, the relevent Bounding Boxes are saved as foreground targets. .

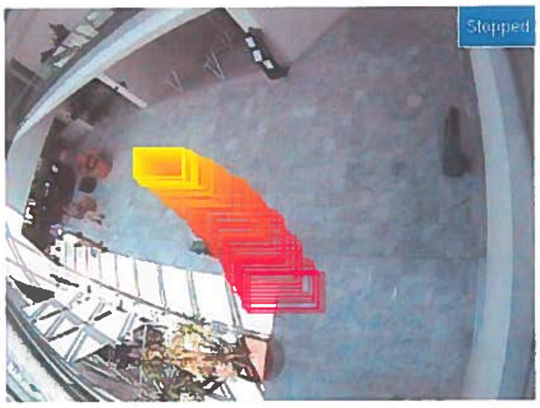
Parameters

The only parameter availible is the threshold to determine the minimum size of a target. Any regions containing less pixels than this threshold value will not be saved as targets.

5.4 Tracking of targets

This algorithm is designed to find overlapping targets in a series of frames, and connect them up into an ordered track. This can represent the movement of a target through time.

the `Track` algorithm is given a cell array of targets as input. Each cell of the array corresponds to a frame number, and within the cell for a particular frame, is a list of targets in that frame. Since the tracking process occurs after target detection is completed, we can easily iterate through the target data, comparing targets in frame `n` with targets in frame `n-1`. The overlap percentage is calculated for each pair of tracks in different frames. If both targets overlap each other by at least the threshold value (in the system, a 20 % overlap threshold is used), then append the new frame to whatever track the previous frame belongs to. If the new target does not satisfactorily match any track in the last frame, then the new target is given its own new track.



Parameters

The only parameter is the overlap percentage threshold previously discussed.

5.5 Track merging

The purpose of this algorithm is to merge together multiple tracks, which are seperated by a few empty frames, to produce longer tracks. Tracks are merged together if their end and start points are similar enough in frame index and location in the image. The merge operation includes filling in missing targets in frames, by interpolating between the location it was last seen and the location it was seen next.

Th input to this algorithm is a list of tracks. Each track is compared to all other tracks in the list. For each comparison, a degree of *joinability* is evaluated. If the two tracks are judged to be *joinable*, then join the tracks together, save the combined track, and delete the component tracks from the list.

The *joinability* is calculated by first ensuring that the tracks do not overlap, and then calculating how close the ends of the tracks are to each other. If the ends of the two tracks are within 10 frames of each other (or similar threshold), then check distances between centrepoints of the ends of the tracks. If the distance is less than 25 (or similar threshold) then the tracks are *joinable*.

The join operation is performed by filling in the missing frames between the ends of the two tracks. This is achieved by finding the coordinates of each target, and then linearly interpolating between the four porner points to create new target locations.

Parameters

The parameters of this algorithm are the minimum frame difference threshold, and the distance threshold, as previously described.

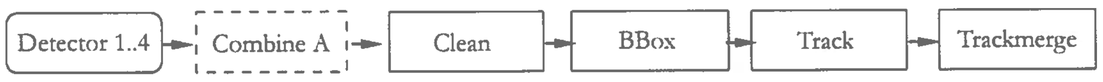
5.6 Combination methods

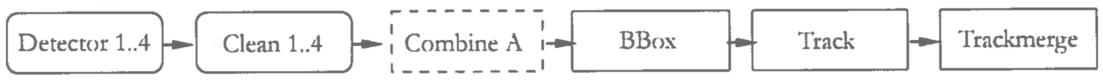
As previously detailed in the System Overview, there are three *Combination Methods* which operate on five possible routes in the system. **`CombineA`** takes binary images as input, **`CombineB`** takes sets of targets as input, and **`CombineC`** takes tracks as input.

5.6.1 Combination method A

`CombineA` takes a set of binary images as input, with each image representing the foreground regions extracted by a particular detection method. These four images are combined together and thresholded to produce a single combined binary image, with combined foreground regions. .

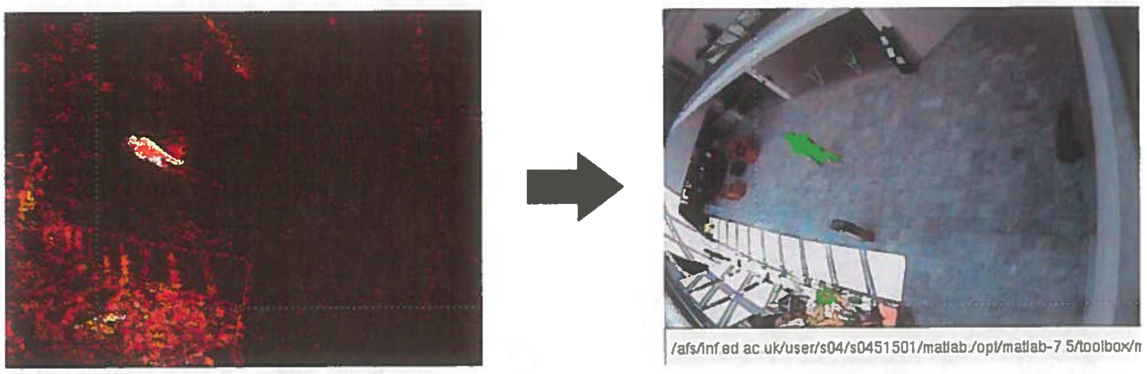
Routes containing combination method





Description of method

Combination method A reads in collection of images, stored in 3 dimensional array, with each value of the 3rd dimension refering to an individual image. Sum the images along the 3rd dimension and then divide by the number of images to get a mean image, where a value of '1' indicates that all the detection methods chose that pixel as foreground, a value of '0.5' indicates that exactly half of the detection methods, and so on. This mean image may be thresholded over a set value, for example '0.3' to produce a combined image.


/afs/inf.ed.ac.uk/user/s04/s0451501/matlab./opt/matlab-7.5/toolbox/n

Choosing values of '1' or '0' for the threshold will return pixels that all the detectors chose, or one-or-more detectors chose respectively.

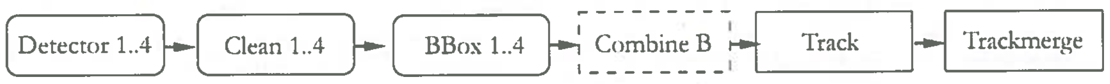
Parameters

The only parameter in this function is the threshold value.

5.6.2 Combination method B

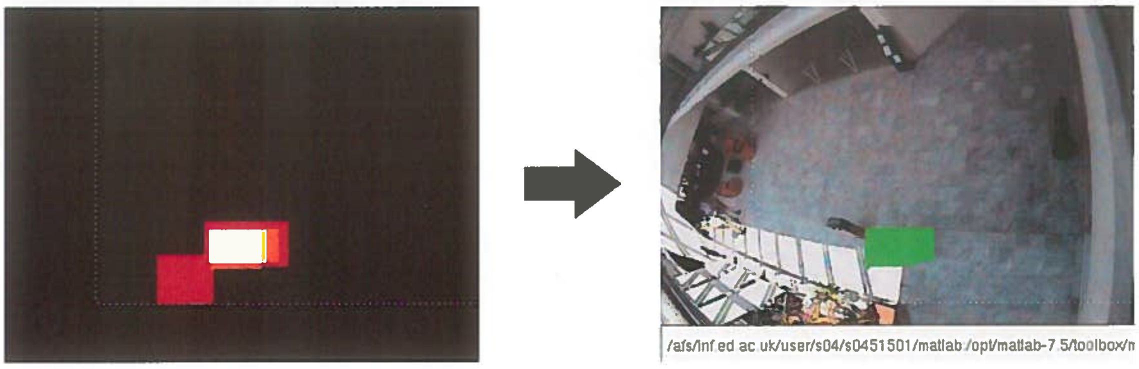
`CombineB` takes multiple sets of targets as input, with each set representing the foreground targets extracted by a particular detection method. These four sets are combined together and thresholded to produce a single combined set of foreground targets.

Routes containing combination method



Description of method

First, a blank combination image is created. The algorithm iterates through sets of targets returned by each detector, and for each detector, creates a blank image. The bounding box coordinates are extracted for each detector, and for all the area covered by target bounding boxes, that area of the image is assigned a value of 1. After performing these operations on all sets of targets, the result is a set of binary images, one for each detector. Combined in a similar way to combination method 1, the pixel values are summed, divided by the number of detectors used, and then thresholded. This produces a binary image of the overlap of the bounding boxes from each detector. The bounding box operation is then run on this foreground image again, as if it were the foreground returned by an individual detection method at the first stage of the system.


/afs/inf.ed.ac.uk/user/s04/s0451501/matlab:/opt/matlab-7.5/toolbox/m

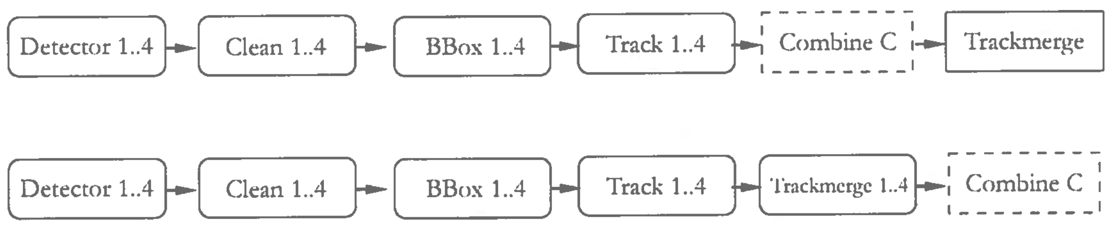
Parameters

The only parameter in this function is the threshold value, and it is most likely approx. the same value as in combineA, if a fair comparison is desired.

5.6.3 Combination method C

Combination method C operates on tracks, and tests if there is overlap between them. If there is a sequence of frames where two tracks have targets in similar locations (with a given distance threshold) then this constitutes overlap. Where overlap occurs, the two tracks are combined into one, which may be longer than one or both of the tracks individually.

Routes containing combination method



Description of method

Combine method C does not compare tracks from the same detector against each other, it only compares tracks produced by seperate detection techniques. The algorithm goes through all the tracks in turn, starting with those produced by the first detector. Each track is compared to all the tracks produced by other methods, and the contents of a 'combined tracks table', which is initially empty. If the two compared tracks are similar then a combination of the two tracks is produced. This new combined track is added to the combined tracks table and the two component tracks are deleted.

Similarity is calculated by function. The first test performed checks if the frame ranges (the start and end frames) of the two compared tracks, overlap by a minimum number of frames (this is refered to in the code as the frame threshold), and if this test is passed, the 'closeness' of the two tracks is calculated by finding an average distance from one another. This is evaluated on each frame by finding centrepoints of each target, and finding the distance between the two using Pythagoras' Theorem. By summing these distances, and dividing by the number of overlapped frames, the mean distance may be calculated. If this distance is below a threshold value then the two tracks are judged to be similar.

The track combination function imputs two overlapping tracks, and outputs a single combination track. The combination is achieved by finding the longer track, and extending it using the shorter track if possible. If one track overlaps the entirety of the other track, then the output is simply the longer track.

Parameters

The parameters of this algorithm are the threshold of the mean distance, and the minimum no. of frames overlap, as previously described.

6. Performance Testing Methodology

6.1 Ground truth and image sequences

To evaluate accuracy of the system, we must measure the system output against actual results. The ground truth is, as was described before, the *gold-standard* of results, and specifically, target locations. We assume that the ground-truth results are of extremely high accuracy, as they were produced by labelling the data manually.

In order to compare the ground truth with the target data, the ground truth must first be imported into MATLAB. This required some work, as the original ground truth files from the *CAVIAR* and *BEHAVE* datasets were encoded in a custom XML language, which allowed many features of the targets to be recorded. The only information needed by our system was bounding box positions however. The data was imported by writing a series of parsers in the Python language, to read in the custom XML file and output a set of comma-separated values files (CSV), which are readable by matlab. This data was then read into Matlab and saved as .mat files, which can be saved and loaded into memory as needed.

All the frames in each sequence were not used, it was decided to limit the selection to five sequences of 100 frames. The sequences were chosen to be relevant to the project, which means studying sequences with only one moving human target to detect, with minimal occlusion, because maintaining identities is not a goal of this project.

Clips were therefore chosen where targets were walking individually. In several cases, other targets were walking or standing in other areas of the scene. It was therefore necessary to be able to select a subset of the ground truth and select a region of an image, so that unimportant target information could be ignored. This is achieved by filtering ground truth results by target ID (where each ID refers to a different human in the scene), and by using functions that limit the area of the image where foreground pixels are detected.

Training frame ranges were also chosen within each data set, to serve as the source of the background model training data for the detecttion methods.

For the *Kernel Density* and the *Mixture of Gaussians* methods, the background model requires a set of history frames to be stored. This process takes several minutes and the resulting data takes up approx. 100 MB (uncompressed RGB

X 50 frames) for each image sequence. So save time, once the background model for a scene has been constructed, it is saved to another .mat file, so that can be easily loaded when needed. Further to this, additional information such as custom parameters are stored on each sequence clip and loaded by the `setsequence(n)` function. The `setsequence(n)` function is executed before any detector can evaluate sequence `n`.

6.2 Measures of accuracy

The ultimate measure of accuracy of the system, is the degree to which a tracked target is 'hit' by the detection process. In some detection and tracking systems, it is important to assign the correct identity to a target. In this case, because only one human target appears in the clips chosen (due to restricted input data), success is judged by how well that single track is covered by the detector, or multiple detectors used.

Whatever route is chosen, the output of the system (as seen earlier) is a set of tracks. It was also shown that a track is represented by a sequence of target coordinates with corresponding frame numbers.

For each target in the detected track, a decision is made on whether it matches the ground truth target in the same frame. This binary classification is performed by comparing the target coordinates from the detection system to the target coordinates in the ground truth, and calculating the percentage of overlap of the two boxes. This percentage overlap is tested against the overlap threshold (both ways, overlap of both targets). The custom parameters loaded by the `setsequence(n)` function, allows custom thresholds for a clip. For example, some clips contain more easily-detectable targets than others. If even the least suitable detection method and parameters can easily match targets, then it is desirable to make targets harder to match. This can be achieved by asking for a higher overlap before a match returns positive.

Analysing a track means: counting the number of True Positives (targets that matched the Ground Truth) and counting the number of False Positives (targets that did not match the Ground Truth. Using True Positives and False Positives doesn't use all the availible information though, as we can also calculate information that tells us how many targets in the ground truth were not detected (False Negatives). This is done by appending a column of '0's to the ground truths data, and if a target is 'hit' by the detection system, that appended cell is set to '1'. For a 100 frame sequence with one ground truth target per frame, the number of False Negatives equals 100 - the sum of all the appended column.

Instead of comparing True Positives and False Positives, we can also use the False

Negatives data by plotting Precision/Recall points.

$Precision = TruePositives/(TruePositives + FalsePositives)$

$Recall = TruePositives/(TruePositives + FalseNegatives)$

The ideal value to return for each is 1. Running the detection system on a sequence with a set of parameters, will give a precision/recall results value. We can run many tests on a detector and get many resultant Precision/Recall values. These can be plotted as points to produce a *Receiver Operating Characteristic* (ROC) curve.

6.3 Testing Parameters

Finding suitable parameters to use during testing must be motivated by the purposes and goals of the system. In our system, the same detector is to be input the five image sequences, and the parameters are to be kept the same for each clip. The goal is to find all-round well-performing values for the parameters of each detector.

To fully test the combination of the detectors, all permutations of parameters and detectors must be computed. That is to say, that all parameters tested on one detector must be combined with all parameters of the 2nd detector, and the 3rd, and the 4th. So if there are p sets of parameters, and d detectors, on i image sequences, and with c combination methods, the number of combinations to compute is cip^d. In the system developed, $c = 5$, $i = 5$ and $d = 4$. So $25p^4$ equals the number of times each detector must evaluate an image sequence of 100 frames.

Therefore it is a prudent to choose a low number of params for each d. There are 3 distinct locations of clips chosen, each with distinct characteristics. Specifically there are three physical locations, the outdoors car park, the mall, and the foyer. In light of this, three well-performing sets of parameters are chosen for each detection method.

So therefore $25x3^4 = 2025$ evaluations of sequences must be performed.

6.4 Optimisation

Evaluation speed is slow with two of the detection methods: *Kernel Density* and *Mixture of Gaussians. Kernel Density* takes approx. 30 seconds per frame! Therefore to save execution time when running all 2025 tests, the foregrounds extracted with these two methods were saved for all parameters, and quickly loaded when the detection methods were to be combined.

$3x5x100 = 1500$ image files were saved for each of the two detectors.

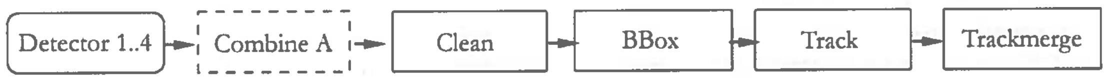
So in testing, all routes are given all 81 combinations of detection parameters, and the result of executing one route of the system will output a set of tracks, as previously described. The frames of each track will be tested against the ground truth, and the mean Precision, Recall value will be computed for that set of tracks.

7. Results

7.1 Routes

This section of the Chapter displays results pertaining to individual routes of the system. For clarity, the original description of each route is repeated here.

7.1.1 Route 1

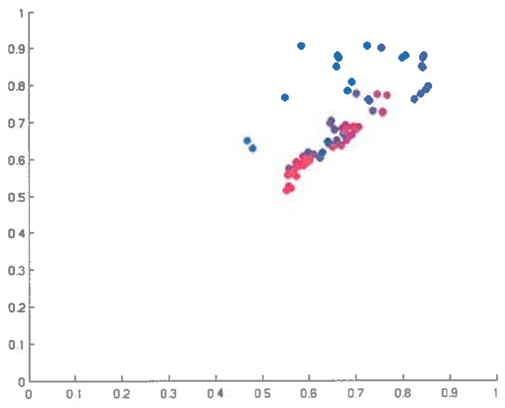


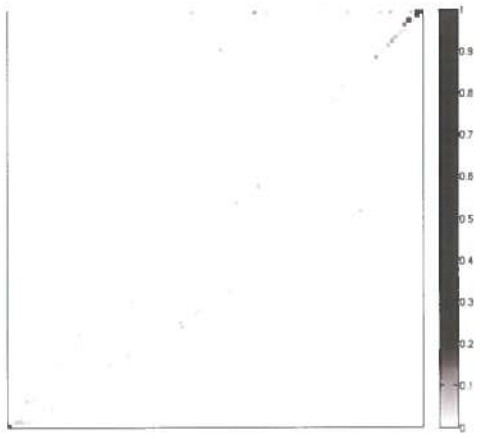
This combination strategy runs the four `Detector` algorithms, returning four binary images containing the foreground pixels from each detector. `CombineA` inputs these foreground images, and combines them to output a single foreground image. The image is cleaned, bounding boxes are extracted, and once the system has completed running these algorithms on a complete set of frames, tracks are created from overlapping frames. `Trackmerge` is finally used to link together the resulting tracks.

The following graphs in this Chapter plot *Precision* on the x-axis and *Recall* on the y-axis. The image on the left plots a point for each combination of detection parameters, by averaging the results for that parameter over all five sequences. The graph on the right is a heat plot which displays all the results obtained by this route. Where points overlap, the colour value is incremented, and a custom *colormap* is used due to the high concentration of results in certain areas of the graph (i.e. $(1, 1)$ and $(0, 0)$).

The table data contains a set of Mean Values of $Precision * Recall$ for each sequence, and the Standard Deviation from the mean. The purpose of evaluating $Precision * Recall$ is to calculate a single number which represents accuracy in target detection. The best possible result is 1, the worst is 0.

What is red/blue?

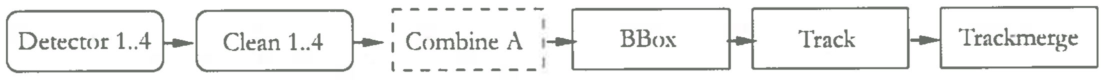




What are axes?

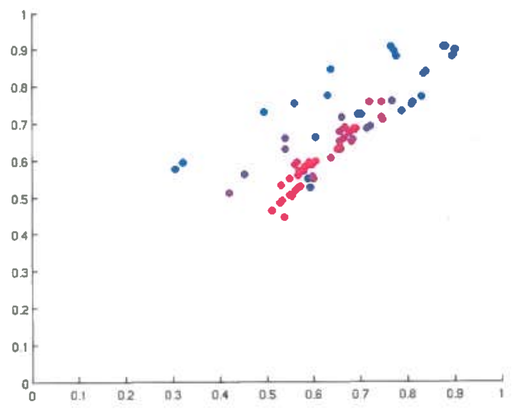
Data Set	Mean Value	Standard Deviation
Sequence 1	0.1235	0.1972
Sequence 2	0.1241	0.1471
Sequence 3	0.8920	0.1675
Sequence 4	0.8594	0.1453
Sequence 5	0.9561	0.0821
Average	0.5910	0.4122

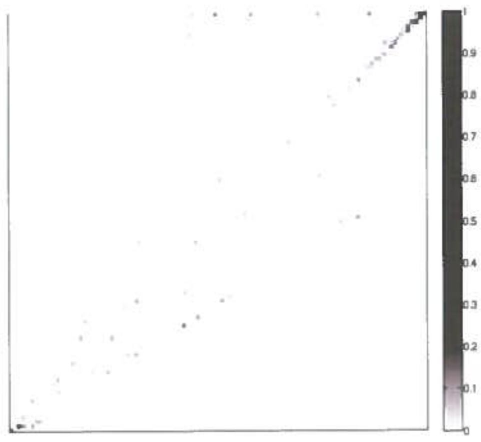
7.1.2 Route 2



This combination strategy is similar to the last, but here, the foreground images from each detector are cleaned individually before being combined by `CombineA` to produce a single foreground image.

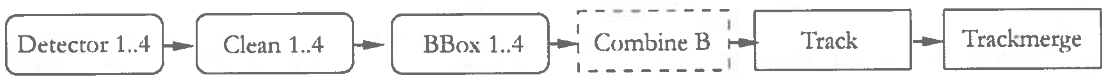
The graphs below plot *Precision* on the x-axis and *Recall* on the y-axis.





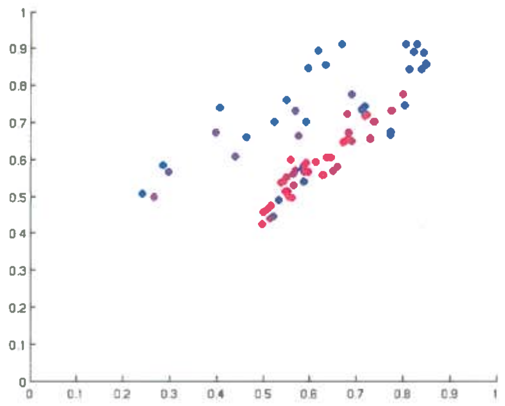
Data Set	Mean Value	Standard Deviation
Sequence 1	0.1300	0.2395
Sequence 2	0.1199	0.1496
Sequence 3	0.8913	0.1625
Sequence 4	0.8062	0.1678
Sequence 5	0.8888	0.1592
Average	0.5673	0.4041

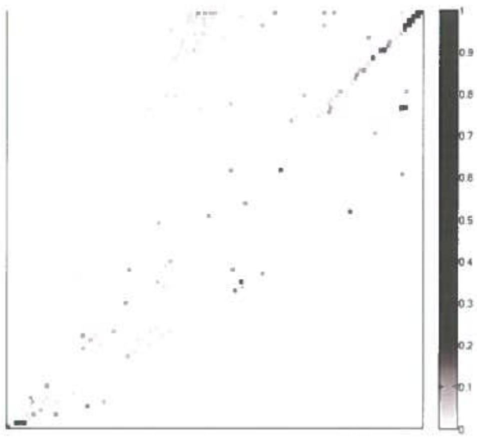
7.1.3 Route 3



This combination strategy combines the data after extracting four sets of bounding boxes, one for each foreground image. This requires a new combination algorithm, `CombineB`, and produces a single set of bounding boxes, which are then tracked and merged.

The graphs below plot *Precision* on the x-axis and *Recall* on the y-axis.





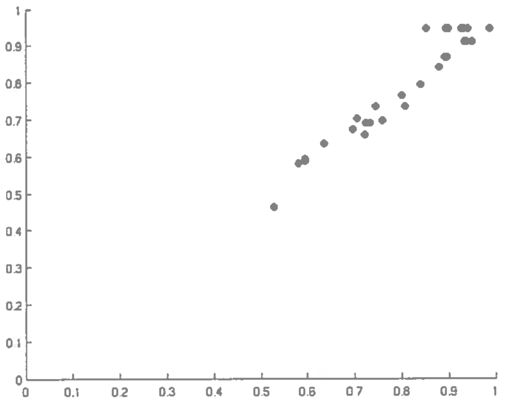
Data Set	Mean Value	Standard Deviation
Sequence 1	0.1448	0.2511
Sequence 2	0.1411	0.1541
Sequence 3	0.8008	0.2151
Sequence 4	0.7493	0.2080
Sequence 5	0.8098	0.1888
Average	0.5291	0.3770

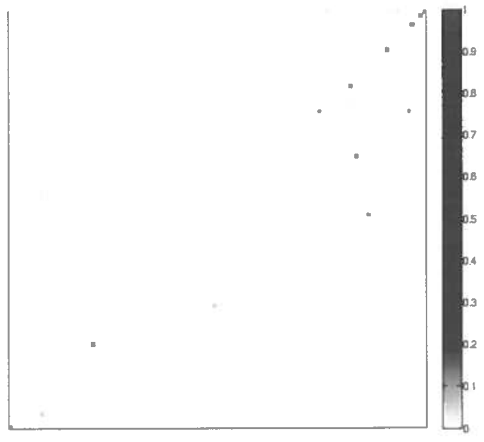
7.1.4 Route 4

$$\begin{tikzcd}
\boxed{\text{Detector 1..4}} \arrow[r] & \boxed{\text{Clean 1..4}} \arrow[r] & \boxed{\text{BBox 1..4}} \arrow[r] & \boxed{\text{Track 1..4}} \arrow[r] & \text{Combine C} \arrow[r] & \boxed{\text{Trackmerge}}
\end{tikzcd}$$

This combination strategy combines the data one stage later again, producing a set of tracks for each detection method, and then using the final combination algorithm, `CombineC`, to produce a single set of tracks which are then merged.

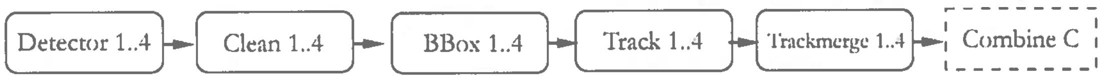
The graphs below plot *Precision* on the x-axis and *Recall* on the y-axis.





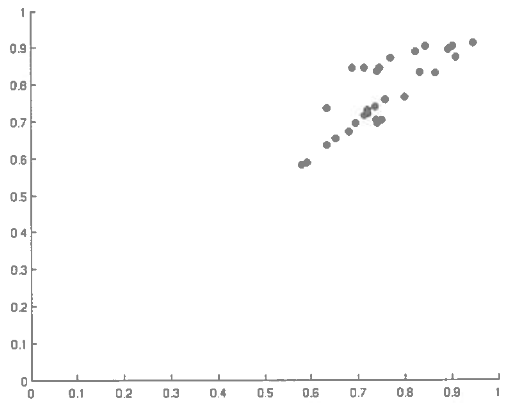
Data Set	Mean Value	Standard Deviation
Sequence 1	0.3676	0.3998
Sequence 2	0.3110	0.2928
Sequence 3	0.9158	0.1827
Sequence 4	0.9230	0.0718
Sequence 5	0.9510	0.0909
Average	0.6937	0.3772

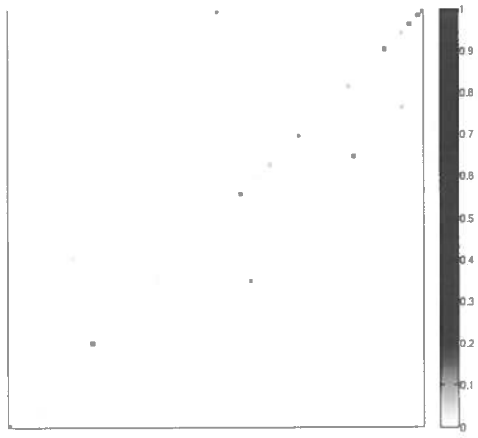
7.1.5 Route 5



The final combination strategy executes the entire tracking operation, up to the `Trackmerge` stage before the combination algorithm is used. As in the previous route, the `CombineC` function is used.

The graphs below plot *Precision* on the x-axis and *Recall* on the y-axis.





Data Set	Mean Value	Standard Deviation
Sequence 1	0.3854	0.3976
Sequence 2	0.2248	0.2405
Sequence 3	0.8729	0.2197
Sequence 4	0.9230	0.0718
Sequence 5	0.9142	0.1755
Average	0.6641	0.3852

7.2 Individual Detection Methods

7.2.1 Static Background Subtraction

Data Set	Mean Value
Sequence 1	0.1384
Sequence 2	0.1028
Sequence 3	0.6735
Sequence 4	0.6120
Sequence 5	0.6975
Average	0.4448

) why so bad?

7.2.2 Frame Differencing

Data Set	Mean Value
Sequence 1	0.0000
Sequence 2	0.0000
Sequence 3	0.1714
Sequence 4	0.4746
Sequence 5	0.2579
Average	0.1808

7.2.3 Mixture of Gaussians

Data Set	Mean Value
Sequence 1	0.1723
Sequence 2	0.2036
Sequence 3	0.4852
Sequence 4	0.5466
Sequence 5	0.3214
Average	0.3250

7.2.4 Nonparametric kernel density estimation

Data Set	Mean Value
Sequence 1	0.0247
Sequence 2	0.0033
Sequence 3	0.6541
Sequence 4	0.3136
Sequence 5	0.6291
Average	0.3458

7.3 Analysis of Results

7.3.1 Comparison of all detection techniques

Detection Technique	Mean Value Over All Data
Route 1	0.5910
Route 2	0.5673
Route 3	0.5291
Route 4	0.6937
Route 5	0.6641
Static	0.4448
FD	0.2579
MixGauss	0.3250
Kernel	0.3458

7.3.2 Discussion

The comparison of all detection techniques results above show all the combination routes implemented to be outperforming any of the individual detection methods. This is a promising statistic, but it is important not to read too much into this, as there is always the possibility that the system implemented has a bias towards a particular function.

When viewing the results of the different routes seperately, it is clear that there is a large disparity between the routes containing `CombineA` and `CombineB`, and the last two routes containing `CombineC`. When looking at the heatmap graph for routes 4 and 5, it is clear to see that there were a large number of results at $(1, 1)$ influencing the overall averages. The apparent advantages of these combination methods may possibly due to unfair advantages afforded by `CombineC`. Specifically, the combination function implemented looks for nearby tracks to combine with one another, and aggressively throws away tracks that do not strongly overlap with another. These features are unique to `CombineC`, and may have affected the results.

When comparing combination methods, it is important not to give advantages to particular methods, as it is only with a level playing field that the superior combination method will be revealed.

8. Conclusions

8.1 What was discovered

It was discovered that it is possible to produce a system that combines image detection methods in a number of ways, and the results will be at least comparable with the component methods. In fact, in the results obtained here, the combination methods produced more accurate results than any of the detection methods did individually.

It was hoped that compelling evidence would point to the superiority of a particular system route, but either due to the parameters and detection methods chosen, or the unfair advantage given to certain routes, no system route emerged as an obviously superior candidate.

8.2 Future Development

There is also great scope for the investigation of more intelligent combination techniques. Currently the methods are simply each given a vote, and this vote is always weighted equally. More intelligent combination techniques would favour background modelling methods based on observed video features. For example if the system detected a large global illumination change, it would be able to select the technique best suited to cope with this.

It is also suggested that combining Detection methods which vary more widely in their background modelling techniques would provide better input data. The more complex methods used here both model a curve over a histogram of the background history, and one detector is unlikely to be able to offer new information for the other. It is postulated that combining edge detection, or gradient based image detection techniques with a kernel based method would provide new information, and provide better data with which to see differences in the combination method more easily.

Bibliography

[1] A. Elgammal and R. Duraiswami and D. Harwood and L.S. Davis *Background and foreground modeling using nonparametric kernel density estimation for visual surveillance.* Proceedings of the IEEE, 2002, pp. 1151-1163.

[2] N. Friedman and S. Russell. *Image segmentation in video sequences: A probabilistic approach.* Presented at the 13th Conf. Uncertainty in Artificial Intelligence, Providence, RI, 1997.

[3] W. E. L. Grimson, C. Stauffer, and R. Romano. *Using adaptive tracking to classify and monitor activities in a site.* IEEE Conf. Computer Vision and Pattern Recognition, 1998, pp. 2229.

[4] S. Jabri, Z. Duric, H. Wechsler, and A. Rosenfeld. *Detection and location of people in video images using adaptive fusion of color and edge information.* Presented at the Int. Conf. Pattern Recognition, Barcelona, Spain, 2000.

[5] The MathWorks *MATLAB.* `http://www.mathworks.com/products/matlab/`

[6] T. Matsuyama, T. Ohya, and H. Habe. *Background subtraction for nonstationary scenes.* In Proc. 4th Asian Conf. Computer Vision, 2000, pp. 662667.

[7] J. Rittscher, J. Kato, S. Joga, and A. Blake. *A probabilistic background model for tracking.* In Proc. 6th Eur. Conf. Computer Vision, vol. 2, 2000, pp. 336350.

[8] B. Stenger, V. Ramesh, N. Paragios, F. Coetzee, and J. Bouhman. *Topology free hidden markov models: Application to background modeling.* In Proc. IEEE Int. Conf. Computer Vision, 2001, pp. 294301.

[9] C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland. *Pfinder: Real-time tracking of human body.* IEEE Trans. Pattern Anal. Machine Intell., vol. 19, pp. 780785, July 1997.

[10] Y.-H. Yang and M. D. Levine. *The background primal sketch: An approach for tracking moving objects.* Machine Vision Applicat., vol. 5, pp. 1734, 1992.