

**An Intelligent CCTV Monitor for Computer Laboratories**

James Matthews

BA Artificial Intelligence & Japanese (Ad Hoc)

2003/4

The candidate confirms that the work submitted is their own and the appropriate credit has been given where reference has been made to the work of others.

I understand that failure to attribute material which is obtained from another source may be considered as plagiarism.

(Signature of student)\_\_\_\_\_

## **Summary**

This project will attempt to create an intelligent CCTV monitor that will track individuals within a computer laboratory. The system will also perform some rudimentary behavioural analysis—categorizing individuals within the lab as moving/standing or sat at terminals.

The project is split into four main chapters: an introduction, background research, system implementation and system evaluation. Appendix B is also used to collect 5 colour plates referenced within the text.

## Acknowledgements

First and foremost, my thanks to James Handley for being an excellent project coordinator. My thanks also to Simon Cogan and Ben Wootton for patiently listening to my ideas throughout the project's lifespan.

Technical assistance from the University of Leeds Vision Group, especially Chris Needham, as well as all at the Generation5 forum (<http://www.generation5.org/forums/>) and local.tex and local.projects.talk on the University of Leeds' Computer Science department newsgroups. A huge '*dōmo arigatō gozaimasu!*' to everybody that supported me when my laptop got stolen one week before this deadline: Clare, Dani, Sam, Dabs, Charlie, Ed and Persa. Finally, many thanks to those of you that introduced me to L<sup>A</sup>T<sub>E</sub>X!

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Aims . . . . .	1
1.2	Objectives . . . . .	1
1.3	Minimum Requirements . . . . .	2
1.4	Methodologies . . . . .	2
1.5	Evaluation . . . . .	2
<b>2</b>	<b>Background Research</b>	<b>3</b>
2.1	Understanding the Problem . . . . .	3
2.2	Object tracking & route learning . . . . .	5
2.2.1	Foreground-Background Differentiation . . . . .	5
2.2.2	Route Learning . . . . .	6
2.3	Kalman Filtering, Motion Prediction & Occlusion . . . . .	6
2.3.1	Kalman Filtering . . . . .	6
2.4	Behavioural Analysis & Hidden Markov Models . . . . .	7
2.4.1	What are Markov Models? . . . . .	7
2.4.2	Applying Hidden Markov Models . . . . .	7
2.4.3	HMM Alternatives . . . . .	8
2.5	Leeds University Vision Group Blob Tracker . . . . .	8
<b>3</b>	<b>System Implementation</b>	<b>10</b>
3.1	Overview . . . . .	10
3.1.1	Modifying the Blob Tracker . . . . .	10
3.1.2	Definitions . . . . .	11
3.2	Computer Lab Model . . . . .	11
3.2.1	Lab Model File . . . . .	12
3.2.2	Preprocessing . . . . .	12
3.2.3	Phase 1: Regioning . . . . .	13
3.2.4	Phase 2: Stacking . . . . .	14
3.2.5	Phase 3: Interpolating . . . . .	14

3.2.5.1	Object initialization . . . . .	15
3.2.6	Phase 4: Removal . . . . .	15
3.2.7	Region Weighting . . . . .	16
3.3	Using the System . . . . .	17
3.3.1	System Options . . . . .	17
3.3.2	System Operation . . . . .	17
<b>4</b>	<b>Evaluation &amp; Future Directions</b>	<b>18</b>
4.1	System Evaluation . . . . .	18
4.1.1	Terminal Usage . . . . .	21
4.1.2	Free Object Detection . . . . .	22
4.1.3	Conclusion . . . . .	24
4.2	Future Improvements & Directions . . . . .	24
	<b>Bibliography</b>	<b>26</b>
	<b>Appendix A: Project Reflections</b>	<b>27</b>
	<b>Appendix B: Colour Plates</b>	<b>29</b>

# Chapter 1

## Project Introduction

---

### 1.1 Aims

The aim of this project is to tackle the large amount of data faced by human CCTV operators. Therefore, the principle goal of the project is to create an intelligent CCTV monitoring program for use within a computer laboratory environment. The program will monitor individuals within the lab and perform rudimentary behavioural analysis.

This has two principle uses: either real-time monitoring of the footage, or examination of pre-recorded, archived footage to find instances of suspicious activity.

### 1.2 Objectives

The objectives of this project are:

- Develop the project using an iterative prototyping design methodology and an object-orientated program structure.
- Develop a blob tracker capable of providing trajectory data of objects within a computer laboratory environment.
- Develop a behavioural analyzer for supplied trajectory data to detect and classify behaviour within the laboratory.

### 1.3 Minimum Requirements

The minimum requirements are:

- A discussion of the problem and project results.
- Implement a tracker that can track objects within CCTV footage.
- Rudimentary behavioural analysis of data. For example, ascertaining whether an individual is sat at a terminal or walking.

The possible extensions are:

- Extend object tracking to attempt to classify objects such as individuals, chairs and computer monitor movement.
- Extend behavioural analysis system to incorporate time-sensitive information and estimates of laboratory population, to more accurately determine atypical results (for example, atypical behaviour is less likely to occur when there few people in the laboratory. Similarly, atypical behaviour will most likely occur in the early or late hours of the day).
- Ensure the system can run at real-time or near real-time.
- Use more advanced object tracking to ensure more accurate trajectory data. For example, using shape descriptors to detect when an individual sits in a chair.
- Use shape descriptors to detect whether an object is being carried.
- Use *a priori* knowledge of the computer lab to aid behavioural analysis and occlusion detection.

### 1.4 Methodologies

At this juncture, the project seems best suited to an iterative prototyping development methodology. If the Leeds University Vision Group Blob Tracker (see 2.5 for further information) proves to be the best approach to take, it will require a degree of tweaking and testing to adapt it to a computer laboratory environment.

### 1.5 Evaluation

The project will be evaluated by manually tracking several individuals through a scene, and comparing them to the tracker results. See Chapter 4 for details.

## Chapter 2

# Background Research

---

Background research for this project was conducted in two phases. The first phase was a broad look at the problem, assessing possible avenues of research to pursue and analyzing the current state of machine vision and CCTV technologies. The first phase looked at object tracking, including foreground-background differentiation and route learning, as well as current behavioural analysis techniques.

By the second phase of research, I had determined that utilizing the Leeds University Vision Group's blob tracker would be a promising avenue to explore, since it implemented many of the lower-level motion analysis algorithms required. Therefore, in order to fully understand the underpinnings of the blob tracker, the second phase honed in on areas that the blob tracker already implemented such as the Stauffer-Grimson background mixture models and Kalman Filtering. In addition to this, behavioural analysis techniques such as Hidden Markov Models and their alternatives were researched in order to ascertain how to augment the tracker further.

### 2.1 Understanding the Problem

Firstly, to highlight the issues surrounding tracking within a computer laboratory, Figure 2.1 illustrates various obstacles that might be encountered.

The initial image has two moving objects, namely the proctor pacing up and down the central aisle, and a student stood up by a workstation. The middle image shows the two individuals and their new positions, and a third party that just entered from a common entry point (the door situated off camera to the top-right).

The final image demonstrates the majority of the obstacles. Firstly, the proctor is now close enough to the camera to sufficiently obscure the student visible in the first two example frames. Furthermore,



Figure 2.1: Computer laboratory tracking example

the student that entered in the previous frame has now double-backed on himself after taking a few steps down the central aisle. Also, a student sitting down has suddenly sprung up, making two moving blobs, corresponding to the student and the chair.

It is also of interest to note that the rather drastic perspective (note the size of the proctor in the first frame compared to the last) might cause problems tracking a blob consistently as one large, growing (or shrinking) blob.

Looking through 45 minutes of initial test footage, a number of conditions occurred within the laboratory that were flagged as potential problems:

- *Periods of inactivity, despite many occupants.* Many people, once sat at terminals, exhibit very little bodily movement.
- *Perspective and occlusion.* Computer lab CCTV does not have the same high vantage point most outdoor CCTV has. Instead, low ceilings and a relatively small room can cause drastic perspective changes within the camera's field-of-view. This means objects nearer the camera could occlude several distant terminals.
- *Movement and occlusion.* As people are sat down, either facing or away from the camera, they often present very little of themselves for any system to detect. Furthermore, movement within the labs of other individuals could occlude their positions.
- *Other objects in close proximity.* Laboratories will have as many chairs as potential occupants. Rolling chairs (such as those in the ENIAC and DEC-10 labs) will often move abruptly as a student sits or leaves. Furthermore, many students fidget when sat at terminals, bouncing their legs or tapping their feet while keeping their body relatively still. This sort of movement would be picked up by a tracker.
- *Screen movement.* Initial research has shown that much of the screen movement (especially the default screensaver) will be detected by the blob tracker, and tracked as an object.
- *Shared terminals.* Many times, two or more people gather around a single terminal.

It was these problems that directed both phases of research.<sup>1</sup>

## 2.2 Object tracking & route learning

Object tracking is the most fundamental of processes for this project. It is imperative that a robust object tracker can be utilized in order to retrieve good trajectory data from people moving around the laboratory.

Object tracking essentially involves locating moving pixels between sequential frames in a video stream. In images where only one object is moving, differential motion analysis (subtracting frames to find changes in pixel intensity) can be used to track an object [9].

In this project though, there will be many independently moving objects. Nevertheless, similar principles can be applied. Differential motion analysis is expanded upon by grouping the pixels into ‘blobs’ which can be given a centroid (rough centre point) and can be tracked as an object [9]. Often object tracking involves extrapolating further data from the pixel groups to ascertain exactly what type of object is being tracked.

Under this project though, the majority of objects within each frame could be safely assumed to be a human although motions such those of chairs and even computer monitors could also be detected. Therefore, to aid the robustness of the tracker, it may be necessary to incorporate additional motion analysis to further classify the objects. For example, one trajectory that diverges into two could be assumed to be a person getting up from a chair.

Sonka [9] provides a good overall introduction to trajectory/motion analysis and object tracking.

### 2.2.1 Foreground-Background Differentiation

Foreground-background differentiation is also a key element in many machine vision systems. If the system can determine what elements lie in the background, analysis of foreground objects becomes much easier. In simple, static scenarios, frame subtraction suffices [9], but most require a more complex method.

Adaptive foreground-background differentiation (or backgrounding) uses an averaging of images over time. This works well in sequences where a large proportion of the image is background, and the foreground is continuously in motion [10]. This is therefore not well-suited to a laboratory environment.

Stauffer & Grimson [10] developed a system of using a mixture of Gaussians to model individual pixels. By analyzing the Gaussian mixtures, the system can differentiate between foreground and background pixels. Furthermore, every time the Gaussians are modified for a given pixel, the system can re-determine whether the pixel is part of the background or foreground using a simple heuristic.

---

<sup>1</sup>Interestingly, many of these traits seem more prevalent in labs belonging to computer science departments, where students are more comfortable in front of terminals. Other, more public labs, tend to exhibit more singular behaviour (terminals further apart, little interaction between students) and more movement at terminals (shuffling paper, more pronounced typing).

### **2.2.2 Route Learning**

Route learning was deemed important to this project, since it would enable the system to learn typical and atypical movement within a laboratory environment. Makis & Ellis [7] provide an excellent introduction to finding paths within video sequences and route learning. The route algorithm presented can be roughly laid out thus:

1. Derive trajectory from tracking an object across frames.
2. Eliminate short, erratic or slow trajectories (such as the double-backing student).
3. Normalize the trajectory to compensate for speed and perspective.
4. Compare new trajectory to existing routes in database using estimated distance.
5. If distance is above threshold  $T$ , create a new route in the database.
6. If distance is below threshold  $T$ , integrate existing path into closest route.

### **2.3 Kalman Filtering, Motion Prediction & Occlusion**

One of the biggest problems foreseen with a computer laboratory tracker is the issue of occlusion. As people move around a computer laboratory, there is a very high chance they will be come partially occluded by rows of computers as they walk behind them. Depending on perspective and camera position, many students will also become completely occluded if they sit down at a workstation.

Initial research found that while there are many methods to compensate for occlusion [12, 5, 8], Kalman filtering provided a solid foundation to aid motion prediction and intrinsically compensates for any occlusion that may occur.

Kalman filtering was not created for the purpose of machine vision and occlusion compensation, but provides an extremely adequate solution for it. By passing trajectory data through the Kalman filter, the tracker can ascertain more accurately where the moving object will be in the next time step. If the object is not present (perhaps due to occlusion), Kalman filtering provides the necessary data to further investigate where the object might be after a further number of time steps. In fact, the Stauffer-Grimson foreground-background differentiation system (see 2.2.1) uses Kalman filtering to help the prediction of blob movement between frames [10].

#### **2.3.1 What is Kalman Filtering?**

Called “probably the greatest discovery in the twentieth century” [4], the Kalman filter solves the problem of estimating the state of a discrete time-controlled process. Maybeck provides an excellent introductory look at Kalman filter, and succinctly describes the process thus:

A Kalman filter combines all available measurement data, plus prior knowledge about the system and measuring devices, to produce an estimate of the desired variables in such a manner that the error is minimized statistically. [8]

Essentially, this can be viewed as a circular process with two stages: time updating and measurement updating. Time updating equations project the current state forward, obtaining an *a priori* estimate of future time steps. Measurement updates provide feedback into the *a priori* estimates to obtain more accurate *a posteriori* estimates [9, 11].

## 2.4 Behavioural Analysis & Hidden Markov Models

An additional step in the project will be to use the trajectory data to perform some rudimentary behavioural analysis. A common method for achieving this is to use Hidden Markov Models (HMMs).

### 2.4.1 What are Markov Models?

Markov models are directed graphs augmented by probability scores. Figure 2.2 shows a very basic Markov Model:



Figure 2.2: Simple Markov Model

Markov Models can be embedded—for example, a speech recognition system might use one Markov Model to ascertain the best phoneme from the given speech data, others might correlate the phonemes to the best word, and a further layer might represent likely sentences containing these words [2].

In actuality, a speech recognition scenario would use a Hidden Markov Model. An HMM extends the principle of a Markov model further by introducing hidden states—states can be separated into observable and unobservable.

Figure 2.3 is a simple representation of an HMM (with probabilities omitted for readability). [1, 3, 2] provide more detailed information on Hidden Markov Models.

### 2.4.2 Applying Hidden Markov Models

In this project, Hidden Markov Models will most strongly be considered to learn the routes and behavioural patterns of people within the computer laboratory.

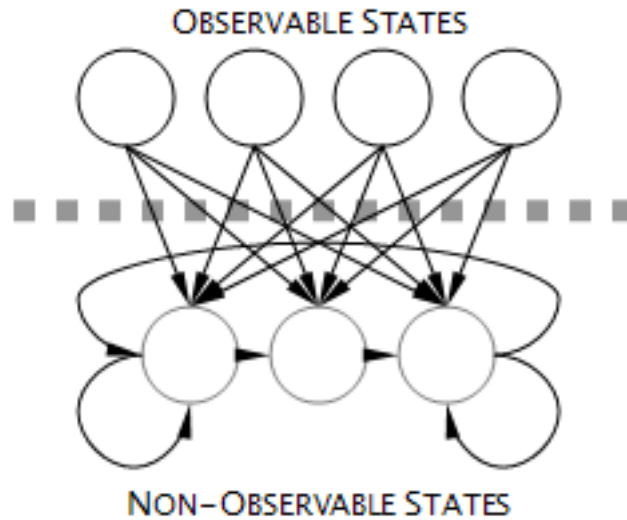


Figure 2.3: A Hidden Markov Model example

Teaching HMMs involves showing the model a variety of test data and “teaching” the model using the ‘Forward-Backward Algorithm.’ Once we have successfully created and initialized the model, we can differentiate between typical and atypical behaviour [1, 2].

### 2.4.3 HMM Alternatives

The techniques discussed have been chosen due to their strong foundations and proven application to the field of machine vision. Nevertheless, alternative avenues have been explored as potential substitute techniques.

The first avenue that will be explored is whether HMMs are in fact overly complicated for a project such as this. It may be sufficient for route data to be gathered, along with the spatial and temporal data gleaned from the object tracker to provide perfectly adequate to perform rudimentary data analysis.

Johnson and Hogg [5] describe a promising use of neural networks to learn the distribution of object trajectories in a given scene.

## 2.5 Leeds University Vision Group Blob Tracker

During the course of researching the project, it became apparent that the blob tracker designed by the Leeds University Vision Group (LUVG) [6] would serve as the ideal foundation upon which to build the project on. The LUVG tracker uses a refined version of the Stauffer & Grimson adaptive background mixture model [10] with Kalman filtering to aid prediction of blob positions across frames (see 2.2.1).

Please see Appendix B, Colour Plate A, B and E for example output of the LUVG blob tracker. The green circles correspond to FGM objects (‘blobs’) while the small red crosses represent initializing FGM

objects. In Colour Plate E, the foreground-background differentiation results have been emphasized and overlaid upon a frame.

## Chapter 3

# System Implementation

---

### 3.1 Overview

This chapter discusses the system implementation, from the initial design of the system through to the actual implementation used. As mentioned, the system was designed to be built upon the LUVG blob tracker, by providing a layer of abstraction between the data from the tracker and any additional behavioural analysis the system had to perform.

As a result of the project's progress, the system itself performs rudimentary behavioural analysis by categorizing whether an individual is standing or moving about the laboratory, or whether they are sat at a terminal.

The remainder of this chapter deals with system specifics.

#### 3.1.1 Modifying the Blob Tracker

The original blob tracker had been created to track much smaller objects, such as cars and pedestrians from high-mounted CCTV cameras. This meant that it would have to be modified extensively to optimize it for a lab environment.

Since the blob tracker follows areas of similar colour (see 2.2.1 for details), it would track individuals walking through the laboratory as anything between one and ten (or more) separate objects. Furthermore, the foreground-background model needed to adapt quicker in order to cope with changes on the user's screen or objects placed upon the computer desks.

Therefore, a further layer of abstraction was required above the output of the tracker. The computer lab model takes the input from the LUVG tracker and attempts to create a more focused model of the current computer laboratory, using a combination of *a priori* knowledge and data averaging.

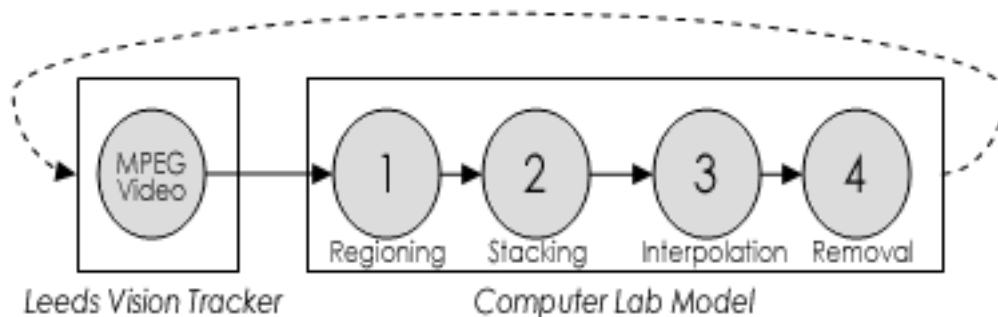


Figure 3.1: An overview of the CLM

### 3.1.2 Definitions

It is important at this juncture to define several terms that will be used throughout this chapter in describing the computer lab model, or CLM. An FGM (foreground model) is a blob that has been passed to the CLM from the blob tracker. The CLM attempts to classify these FGMs into two categories, *regioned objects* or *free objects*, with the aid of a *lab model file*.

- A *lab model file* is a simple, flat-text list of regions within the CCTV camera's field of view that corresponds to terminals within the laboratory.
- A *regioned object* is defined as an object that lies totally within a region defined in the lab model file. Similarly, a *regioned area* is simple an area defined by the lab model file.
- A *free object* is defined as an object that is present or moving outside of the defined regioned areas.

During the course of the chapter, we will also talk about *potential objects* and *initializing objects*.

- A *potential object* is an object that the preliminary phases (1 & 2) have flagged as a candidate object for inclusion or interpolation within the CLM.
- An *initializing object* is an object that has been included into the CLM in the interpolation phase as a new object. It is not displayed or logged by the system, until it has been reinforced by the weighting system (see 3.2.5.1).

## 3.2 Computer Lab Model

The Computer Lab Model is the fundamental part of the project—turning the output of the foreground model into usable data within a laboratory environment. During the course of the project, the CLM structure was entirely rewritten. The initial CLM processed the data from the blob tracker and attempted

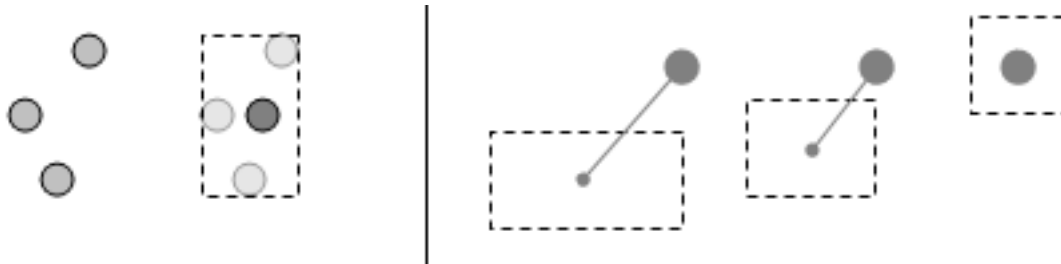


Figure 3.2: Foreground models are averaged and have an associated bounding box created for a corresponding CLM object

to correlate as much information as possible before applying the *a priori* knowledge. The effect of this was that moving objects were tracked with promising results, but people sat at terminals were increasingly difficult to monitor. Furthermore, objects that transitioned from a stationary position at terminals to walking or standing were often tracked by multiple objects due to the heuristics involved in merging the averaged data and *a priori* knowledge.

A major revision of the CLM essentially reversed the process; applying the *a priori* knowledge to the data first and using it to categorize as much output from the blob tracker as possible. The CLM then classifies as much of the remaining data as a standing/walking person. This revision of the CLM also added region weighting to increase the accuracy of laboratory population estimates.

The entire system is an iterative process (across each frame of video) that attempts to refine the data as it receives additional information from the LUVG tracker. The next subsections will look at the lab model file format and the core algorithm's five data-processing phases: preprocessing, regioning, stacking, interpolating and removal.

### 3.2.1 Lab Model File

The lab model file is a simple representation of terminal regions within the camera's field of view. Firstly, the lab model specifies the number of regioned areas used followed by the state and region dimensions for each area. In this system, regions are defined using their top x and y coordinates and their width and height.

Region states are used to help remove noise and cope with camera perspective. A state can either be sensitive (0x00001000 or 4096), generic (0x00002000 or 8192) or insensitive (0x00004000 or 16384). The states are used in regioning phase (see 3.2.3) to detect movement within regioned areas.

### 3.2.2 Preprocessing

The preprocessing phase of the system is concerned with altering objects already stored within the CLM. Objects stored within the CLM have their bounding boxes translated and shrunk toward the true centroid of the object.

Within the CLM system, the bounding box is constructed during Phases 1 and 2. Phases 1 and 2 average data collected from the blob tracker (see relevant sections below for more details), and group similar ‘blobs’ into objects. The bounding box of these grouped blobs becomes the CLM object’s bounding box and similarly for the group’s centroid. *But*, there are constraints on the width and height of the box. As an individual moves within the laboratory, obviously so does the area within the bounding box, and since the FGM data is not stored between frames, only the resultant CLM object, the system has to compensate.

The system therefore restricts bounding boxes to a maximum of 30 pixels in height and width. This method allows the system to track an individual when moving, despite the noisy data from the blob tracker between frames. It has further advantages explained the Removal phase (see 3.2.6).

This has a side-effect, often causing the object’s centroid and the centroid of the bounding box to be severely offset. Fortunately, this works extremely well to the benefit of the system, providing a rough vector of movement within the laboratory and also aiding with occlusion detection (see 3.2.6) and region weighting (see 3.2.7).

It is with this bounding box maintenance that the preprocessing phase is used. Before each frame, the algorithm attempts to move the bounding box slightly toward the CLM object’s centroid, as well as gradually shrinking it to a 15x15 pixel block. This helps by allowing the bounding box to represent the moving object’s velocity (albeit extremely roughly) and also greatly aids detecting transitions between free objects and regioned objects (see 3.2.6 for more information and Colour Plate C for an example image sequence).

The preprocessing phase is also used to decrement the weights of regions (see 3.2.7) and objects (see 3.2.5.1).

### **3.2.3 Phase 1: Regioning**

The first phase groups the foreground models (FGMs) into regions as defined by the lab model file (see 3.2.1). FGMs that are located in the same region have their positions averaged to produce a centroid for a potential CLM object. The object also has a bounding box created from the minimum and maximum positions of the FGMs that constitute the object (see Figure 3.2).

Note that having an FGM present within a regioned area *is not sufficient* for a potential CLM object to be created. Instead, the region state also has to be accounted for. For example, many CCTV cameras will exhibit a rather sharp perspective change, so terminal regions in the foreground, nearest the camera, will produce a lot more movement than background terminals. Therefore, the CLM accounts for this by requiring insensitive regions to have three or more FGMs present within it for a potential CLM object to be created. Generic regions must have at least two, and sensitive regions only require one FGM.

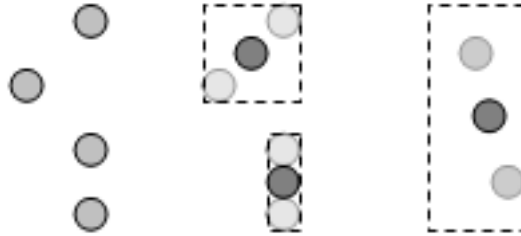


Figure 3.3: A demonstration of how the CLM will stack multiple pairs of FGM blobs

### 3.2.4 Phase 2: Stacking

The second phase attempts to find people that are moving or standing within the laboratory environment. Since the LUVG blob tracker ‘stacks’ several FGMs together for people standing or walking (see Colour Plate B), the CLM attempts to find these stacked areas and assign them one centroid and bounding box in the same fashion Phase 1 does for regioned objects.

The CLM cycles through all FGMs and groups objects that are stacked upon each other, providing at least one FGM lies outside a region. The rationale for this is that people standing and moving about the laboratory will occasionally occlude others sat at terminals, therefore FGMs present within regioned areas cannot be discounted, but if both FGMs lie within a regioned area, Phase 1 will have already grouped and classified them as regioned objects.

Although Phase 2 only initially looks at pairs of stacked FGM objects, it will merge them with potential CLM objects that are also stacked upon the pair. See Figure 3.3 for a clearer representation of this.

### 3.2.5 Phase 3: Interpolating

The previous two phases are purely concerned with the data in the current frame. The interpolation phase applies the knowledge garnered from the previous two phases into the current CLM and updates the lab model object bounding boxes and centroids accordingly. While phases 1 and 2 have been using the FGM data and creating objects for the CLM to use, they are all merely *potentials*. They could represent objects already present within the CLM, objects that are moving, or new objects to be added.

For the first iteration of Phase 3, all potential objects are simply copied to the lab model—creating a rough representation of the lab. In subsequent iterations, the CLM will compare objects in the current frame with the state of all objects within itself. Firstly, regioned objects are compared and have their positions and bounding boxes updated accordingly. Next, potential free objects in the current frame are compared with the CLM’s free objects. If the object lies within a certain distance, it is interpolated and the model has the centroid and bounding box updated. In this implementation of the system, this distance is 40x25 pixels. This stretched window accounts for the bias toward fast horizontal movement within the lab.

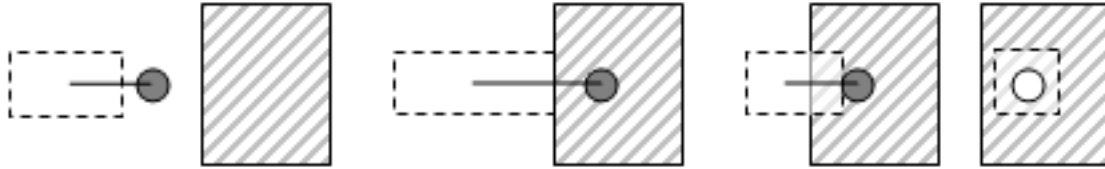


Figure 3.4: A demonstration of how a free object might transition to regioned object. Note that the object does not change state until the entire bounding box is contained within the region.

Any objects that cannot be identified during these two sub-phases are added as new objects, and are flagged as *initializing*.

### 3.2.5.1 Object initialization

Objects are not immediately classified as concrete objects to be added to the CLM, since many might be noise, the results of movement within a terminal monitor (such as scrolling a window or a screensaver), chairs or a variety of other factors. Therefore, the CLM assigns a weight to each object upon entry into the scene.

Each time Phase 3 detects that the object has been used again (through the detection of reused regioned or free objects described above), it is “touched” and has its weight incremented. At the beginning of every frame, all objects within the CLM have their weights decreased (see Preprocessing). The increment weight should be significantly greater than the decrement weight, so that objects do not decay too rapidly. Within this implementation of the system, objects are weighted according to whether they are initializing (+5), regioned (+75), or free (+10).

If an object’s state is initializing and the weight is incremented above a certain threshold (20 for free objects and 100 for regioned objects), the CLM will classify the object as concrete and the system will begin to log the object’s activities. Note that regioned objects are weighted more heavily than free objects, since regioned objects (students sat at terminals) will generally move less and are more reliable between frames.

Weights for initialized objects are also capped at 1000 for regioned objects and 200 for free objects.

### 3.2.6 Phase 4: Removal

The final stage involves removing objects that are no longer required. This involves either objects whose weights have dropped to zero, or free objects that are completely contained within a regioned area.

A object whose weight has dropped to zero will often correspond to a student who has walked out of the laboratory. The weighted system will also allow objects that are moved momentarily within the laboratory (such as chairs) to be removed from the CLM quickly once motion has stopped.

The transition from a free object to a regioned object will occur when, for example, a student enters the lab and sits at a terminal—the free object will “become” a regioned object. Due to the way the interpolation phase will expand the CLM object’s bounding box and the subsequent shrinking and translation of the bounding box toward the object’s centroid at the beginning of each frame (see Preprocessing), once movement has stopped, the object centroid will converge upon the object within about a second. If the object, and all its entire bounding box are contained within a region, the free object is transitioned to a regioned object. Figure 3.4 shows an example scenario. Colour Plate D also shows an example of a free object passing through a regioned area.

### **3.2.7 Region Weighting**

As mentioned, the second version of the CLM features a region weighting system. This weighting system is similar to the object weight system described in 3.2.5.1, whereby each region is assigned a weight that is controlled by the amount of movement occurring within it.

This was designed to counteract the lack of movement within regioned areas. As students move into the lab, the CLM tracks them as free objects. Once they have sat down, the free object will transition to a regioned object. The object will continue to track while the student is still moving. Once a student has settled down and starts typing at the terminal, virtually all body movement will stop. This causes the CLM object to disappear after several seconds since its weight drops to zero and Phase 4 removes it from the model.

Therefore, during Phase 3 any regioned object ‘touches’ its associated region, increasing its weight. During the preprocessing phase, each region has its weight decreased slightly. Again, increase weight should be significantly greater than the decrease weight. In this implementation, weight decreases by one each frame and is increased by 30 to a maximum of 1500 if movement is detected.

The result of this region weighting is that the system continues to successfully determine that the terminal is occupied even with relatively small movements. Any movement at any point by a regioned object will cause the region’s weight to spike. Note that only initialized, regioned objects can touch an object. Initializing objects should not be allowed to influence region weighting since they can be the result of noise, and free objects cannot be allowed to influence region weighting since they may be associated with objects moving past the regions (as explained in Phase 4).

The only downside to this system is the length of time the system continues to detect a person sat in the terminal, despite them being absent. If a student truly does stand up and move away from the terminal, the motion within the region will often cause the region weighting to spike to its maximum allowed value (1500 in our system), therefore taking approximately 30 seconds to drop to zero and be flagged as empty.

Colour Plate D demonstrates this. Observe the central terminal: its weight is decreasing but is not low enough to be flagged as inactive.

## 3.3 Using the System

To use the system simply type:

```
./cctv <options> <video file>
```

The video can be either in either MPEG or Quicktime format. The resolution should be around 360x240 (test footage was 352x290), since it affects the performance of the blob tracker and the expected size of the objects tracked.

### 3.3.1 System Options

The system has a variety of options to help visualize or debug the results:

- *-fgmodels* - Set to either 0 or 1 to show the output from the blob tracker itself.
- *-labmodels* - Set to either 0 or 1 to show the output from the CLM.
- *-regions* - Set to either 0 or 1 to display the regions listed in the lab model file. The regions will be displayed in a gradient between orange and white. A completely orange region represents an unused region, while a white region displays an active region.
- *-boundingboxes* - Set to either 0 or 1 to display the bounding boxes around the CLM objects.
- *-pauseafter x* - Will after x frames.
- *-slowmo x* - Where x is between 0 and 4. 0 will run the system at full speed, while the others will gradually decrease the speed of the system.
- *-labmodel* - Set to the lab model to be used.

### 3.3.2 System Operation

While the system is in operation, it will output the movement of free objects, as well as variety of other information regarding region usage. The main output of the system is the overlaid upon the CCTV footage. Green circles correspond to FGMS, blue circles to free objects, white circles to regioned objects and red circles to initializing objects. Furthermore, bounding boxes are shown with vectors connecting the object's centroid to the centre of the box. Regions can also be shown with their region weights depicted as a gradient between orange and white.

Once the system is running, you can briefly display region information by right-clicking in one of the system windows. Furthermore, holding down the control key and left-clicking one of the windows will cycle through the five speed settings. Left-clicking will display the x and y position of your mouse, the frame number and the region number (-1 for no region).

## Chapter 4

# Evaluation & Future Directions

---

### 4.1 System Evaluation

Overall, the system performed very well albeit in a different fashion to that originally intended. The behavioural analysis portion of the code became an intrinsic part of the computer laboratory model, instead of a separate layer as originally envisaged.

As part of the iterated prototyping methodology used, the system was continually evaluated. As most parameters were set using empirical testing, much of the very early evaluation centred around the visualized results. Once the project had progressed, the initial program evaluation essentially came in the form of assessing its data filtering and continuity. An excellent example of this is to look at Figures 4.1 and 4.2.

Figure 4.1 shows the CLM's output for *one* object in 2D space (the position of the object as represented on the video screen). It uses the first version of the CLM, and you can see that while it is doing a good job of tracking objects in each frame (as shown by the clusters of detected objects), it does nothing to keep continuity across frames. If it did, there would be one large group clustered around a single region, not the five clusters with about twenty additional smaller hits as exhibited.

Figure 4.2, on the other hand, uses the latest version of the CLM. The graph is a 3D representation of four selected objects, with their X and Y positions plotted over time (frame number). You can see how the CLM now keeps continuity across frames, using the same CLM object despite lapses in detected movement.

Once the data filtering was perfected, the system had to be evaluated as a whole. As mentioned, extensive behavioural analysis could not be performed due to the overly noisy output from the LUVG blob tracker, so applications such as hidden markov models (see 2.4.1) and route learning (see 2.2.2)

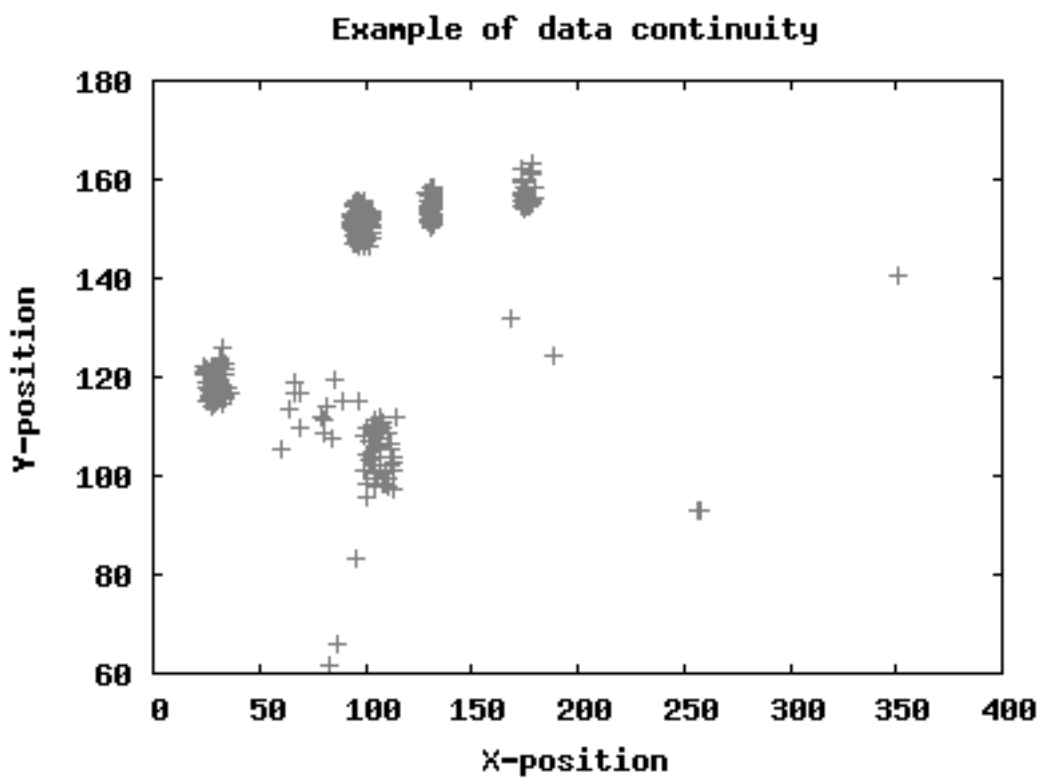


Figure 4.1: An example of one object's position on the video screen as represented within the CLM. The graph was created using a halfway build of the code. Notice how the object, while locally forming groups, jumps about the screen a great deal. The same object in the CLM would not represent the same object within the world.

### 3D Representation of Terminal Usage over 15 Minutes

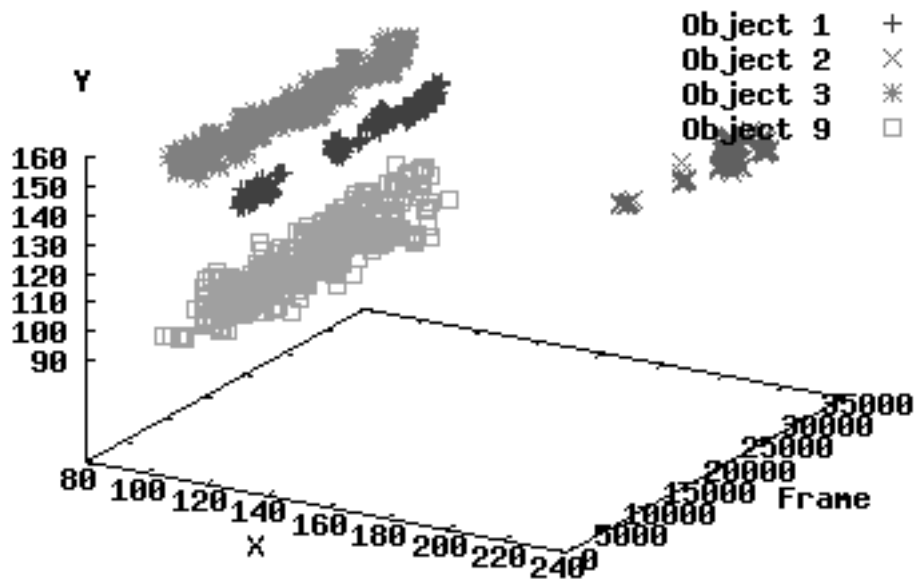


Figure 4.2: A 3D graph of a few objects tracking successfully using the final version of the code. Notice how multiple objects are tracked continuously, and by the same CLM object despite occasional lapses in detectable movement

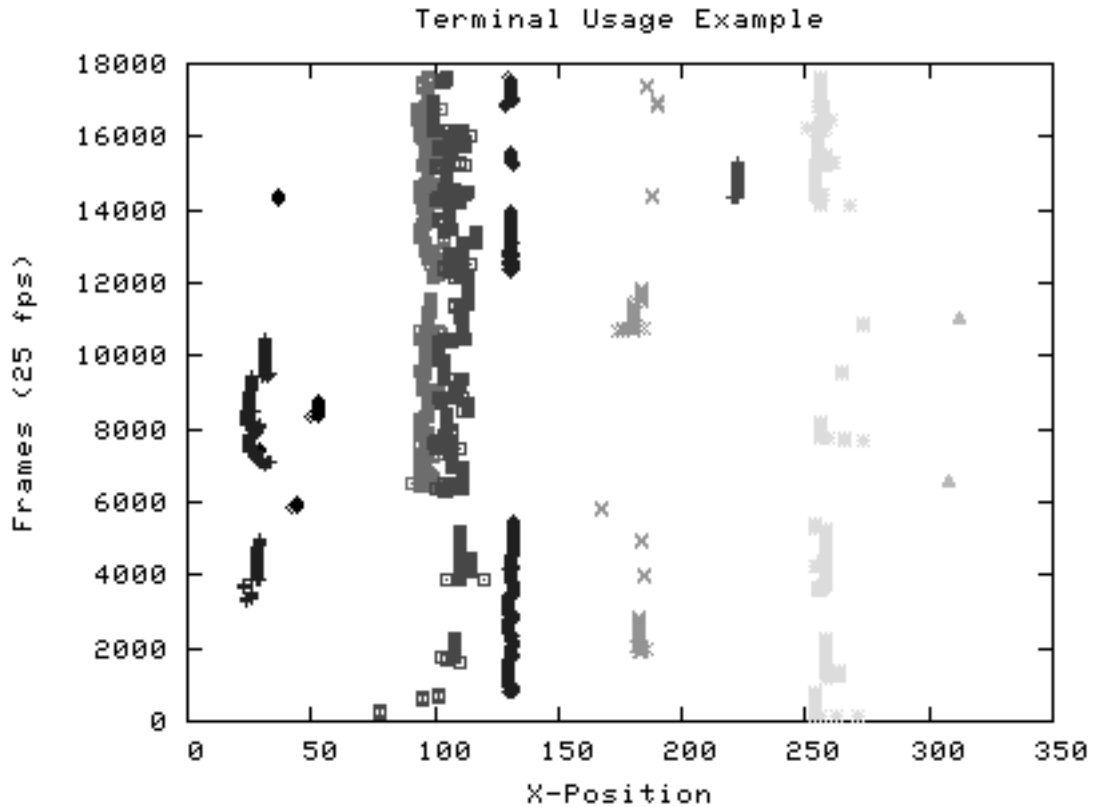


Figure 4.3: Determining terminal usage using positional and CLM-only data

could not be utilized. Nevertheless, using a lab model allowed the system to analyze terminal usage and laboratory population to a certain degree of accuracy.

Therefore, the system evaluation centred around how well the system performed at tracking individuals about the laboratory. Due to the way the system actually tracked individuals, with lagging bounding boxes as well as the two types of tracking (free and regioned), it was decided that the best measure of the system revolved around its knowledge of terminal usage and laboratory population.

#### 4.1.1 Terminal Usage

Terminal usage is important to the system, since it helps greatly toward laboratory population estimates, as the majority of people in a computer lab will be sat down. Furthermore, being sat at a terminal is acceptable behaviour for a computer laboratory, and can therefore be used to quickly ascertain which individuals require further behavioural analysis (if such analysis existed in the system).

Figure 4.3 shows the output of the system for all regions over nearly 7 minutes of video footage during a relatively busy time in the labs. The x-axis represents the x-position of the CLM regioned object for each of the defined regions in the test footage, while the Y-axis the frame number.

The graph clearly displays the system detecting the regioned objects, but if you look carefully you

can see how all the detections come around periods of slight horizontal movement within the region itself. Once the individual has stopped moving, the system will eventually remove the object.

This sporadic detection can be combated by using the region weighting.

Using the region weighting system, regions will still be flagged as active, despite the lack of an accompanying CLM object. If the terminal user moves sufficiently once a minute, the CLM will maintain a fairly accurate count of the laboratory population.

To evaluate the region weighting and the rest of the system more thoroughly, a more extensive test was performed. Every 30 frames, a sample was taken of the number of free objects and regioned objects present in the system. A human operator also recorded the number of regioned and free objects they detected. Figure 4.4 shows the results of this test. The y-axis shows the detected number of objects, the x-axis represents time.

The graph shows some interesting results. Firstly, it is important to note that the system is presented with the video stream *as-is*. It is not told anything about the number of people already present in the lab, before it starts to analyze the footage. Therefore, it takes a while for the system to accurately judge the number of people in the lab, since they all need to move to be detected. This is seen by the stepped nature of the ‘detected region’ data. It is worth noting that once the system has ‘warmed up’, it matches the human operator’s numbers exactly.

These numbers could sometimes be misleading, since the system will falsely detect one region as active (normally due to a recently vacated terminal, whose weight has not yet dropped to zero) and will not detect another terminal (normally due to a particularly motionless occupant). This scenario occurred once during the test. Nevertheless, the overall system performance for regioned object detection (warm-up period aside) is extremely promising.

### 4.1.2 Free Object Detection

Free object detection remains the most difficult part of the project. While free object detection works, it is still not reliable enough nor does it discriminate between objects, yielding several false detections.

Again, looking at Figure 4.4, you can see that the system always detected a free object when the human operator did, but it often detected additional objects that the operator would ignore. In the test data, the majority of the small bumps in the graph coincide with chair or leg movements—significant movement that is triggered by the regioned individual, but occurs outside the regioned area.

The biggest problem with the free object routines can be shown by the large spike (5 free objects detected). This spike occurred in the test footage when two individuals quickly walked toward (and eventually beneath) the camera. This very sharp, fast change in perspective and object size caused the system to assign multiple free objects to the two individuals.

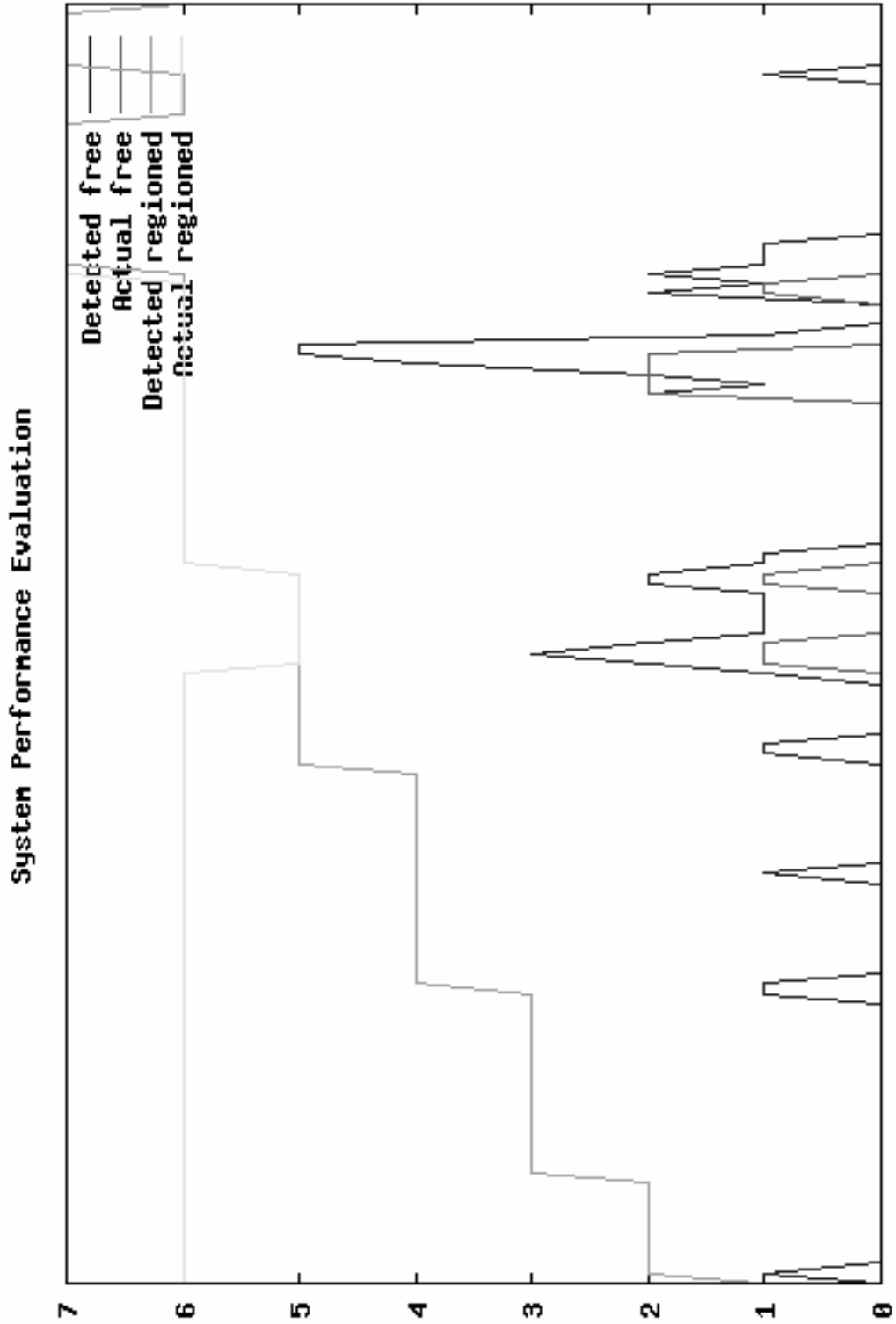


Figure 4.4: Laboratory population using region weighting

### 4.1.3 Conclusion

Overall, the system performs extremely well especially considering the output from the blob tracker. The weighting system of the CLM allowed objects to reaffirm themselves across frames, maintaining continuity and data integrity, and allowed regioned areas to be flagged as active despite a lack of detectable movement within their areas.

Free object tracking would successfully track any individual moving around the lab, but would sometimes falsely detect other objects such as chairs or leg movement beneath terminal desks. The free object routines also had more trouble with perspective than the regioning routines, since they had region states to counteract these effects.

State transitions worked well, using the lagging bounding box system. The system would successfully transition a free object to a regioned object using a bounding box as it shrunk and translated toward the object's centroid. This same lagging bounding box technique ensured that free objects would not be falsely transitioned unless they sat at a terminal.

While there is still a great deal of improvement to be made, the system can successfully monitor quite complex movement in a unique environment—coping with both long, quiet periods and short, busy bursts of activity.

## 4.2 Future Improvements & Directions

While the system performs adequately, numerous improvements could be made:

- *Use more advanced object detection.* While the blob tracker was perfectly suited to this project due to time and scope restraints, a more advanced foundation for the project would have yielded much better results. Using shape descriptors would enable the system to make much more intelligent guesses as to the object's state (sitting down, standing up).
- *Adaptive regions.* Without the use of shape descriptors, adaptive regions might provide slightly more accurate results. Regions could move and expand/contract slightly according to surrounding movements. The system sometimes had trouble when individuals sat slightly misaligned in a region—for example, sitting far back from the screen to type, or sharing a terminal with a friend.
- *Complete behavioural analysis.* With better object trajectory results, behavioural analysis could prove a very real possibility. Additionally, time information as well as lab population could be used.
- *Improved visualization tools.* Gawk and Gnuplot were used to extrapolate the logged data and create graphs and diagrams to aid system understanding. Integrated visualization tools would be an asset to both the system developer and the end-user.

A few smaller 'todo'-style improvements could be made:

- *Expand lab model file format.* Many of the parameters and video resolution-specific variables could be moved out of the program into the lab model file format itself.
- *Maintain constant ID between states.* Due to the way IDs are assigned and objects are removed and inserted during the removal phase, object IDs are not constant across state transitions. Ensuring they remained constant would mean individuals could be continuously tracked from the moment they walk in, during the time they are at a terminal, and the route they leave the lab by.
- *Improve system interaction.* The system would greatly benefit from a method of allowing end-users to weight regions themselves. This will greatly reduce the ‘warm-up’ period detailed above.

# Bibliography

- [1] R. Boyle. Introduction to hidden markov models, November 2003. [http://www.comp.leeds.ac.uk/roger/HiddenMarkovModels/html\\_dev/main.html](http://www.comp.leeds.ac.uk/roger/HiddenMarkovModels/html_dev/main.html).
- [2] J. Černocký. Hidden markov models - an introduction, November 2003.
- [3] R. J. Elliot, L. Aggoun, and J. B. Moore. *Hidden Markov Models: Estimation and Control*. Springer-Verlag, New York, 1995.
- [4] M. S. Grewal and A. P. Andrews. *Kalman Filtering: Theory and Practice*. Prentice-Hall, New Jersey, 1993.
- [5] N. Johnson and D. Hogg. Learning the distribution of object trajectories for event recognition. In *British Machine Vision Conference Proceedings*, 1995.
- [6] D. R. Magee. Tracking multiple vehicles using foreground, background and motion models, 2001. [ftp://ftp.comp.leeds.ac.uk/scs/doc/reports/2001/2001\\_21.pdf](ftp://ftp.comp.leeds.ac.uk/scs/doc/reports/2001/2001_21.pdf).
- [7] D. Makris and T. Ellis. Finding paths in video sequences. In *British Machine Vision Conference Proceedings*, pages 263–272, 2001.
- [8] P. Maybeck. *Stochastic Models, Estimation and Control*, volume 1. Academic Press, London, 1993.
- [9] M. Sonka, V. Hlavac, and R. Boyle. *Image Processing, Analysis and Machine Vision*. Brooks/Cole, California, 1998.
- [10] C. Stauffer and W. E. L. Crimson. Adaptive background mixture models for real-time tracking. Technical report, The Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1991.
- [11] G. Welch and G. Bishop. An introduction to the kalman filter. Technical report, Department of Computer Science, University of North Carolina, 1995. TR95-041.
- [12] M. Xu and T. Ellis. Partial observation vs. blind tracking through occlusion. In *British Machine Vision Conference Proceedings*, pages 777–786, 2002.

# Appendix A: Project Reflections

---

I feel that the project has gone extremely well, despite moving in a slightly different direction to that originally envisaged. It has definitely given me a taste for computer/machine vision that I hope to continue in postgraduate studies.

It has been interesting working with third-party code (the LUVG blob tracker and real-time image library) so extensively—I have definitely learnt the value of well-documented and structured code first-hand.

The methodology approach I choose at the beginning of the project also paid off greatly, since I had to approach the project initially from a learning perspective. The LUVG code is large and getting to grips with the parameters, the output, and the code itself was a steep learning curve, and a challenging way to start the project. I approached this by playing about with the LUVG output and attempting to visualize a variety of experiments, using the iterative prototyping methodology to tweak and build upon my last experiment. The cumulative result of these experiments was the initial version of the CLM.

Eventually though, it was apparent a different approach was needed. Again, taking the iterative prototyping stance, I took the code I had and thoroughly examined it to understand what I might do to significantly better it. I came to the conclusion that essentially reversing the process was the best way and restarted the coding process from scratch.

The result of this methodology meant that I not only achieved the minimum requirements, but two of the additional objectives—namely, to run the system at real-time (the system actually runs faster than real-time on the standard DEC-10 machines), and the use of an *a priori* model of the computer laboratories.

I think my initial goals for the project were a little ambitious, coupling behavioural analysis with a machine vision system is a complex procedure both to implement and evaluate. I think if I had realized this during my background research phase, I could have concentrated more on the machine vision system rather than focusing on the behavioural analysis aspect as I did. Nevertheless, the research did give me an excellent understanding of where the project should progress—the system was designed to provide

accurate data to a behavioural analyzer from the beginning, whether it was integrated within this system or not.

I've learnt a great deal from this project: the intricacies of computer lab behaviour, a firm understanding of how real machine vision systems work and a sound grasp of modern behavioural analysis techniques. Two out of three of those should serve me extremely well in the future.

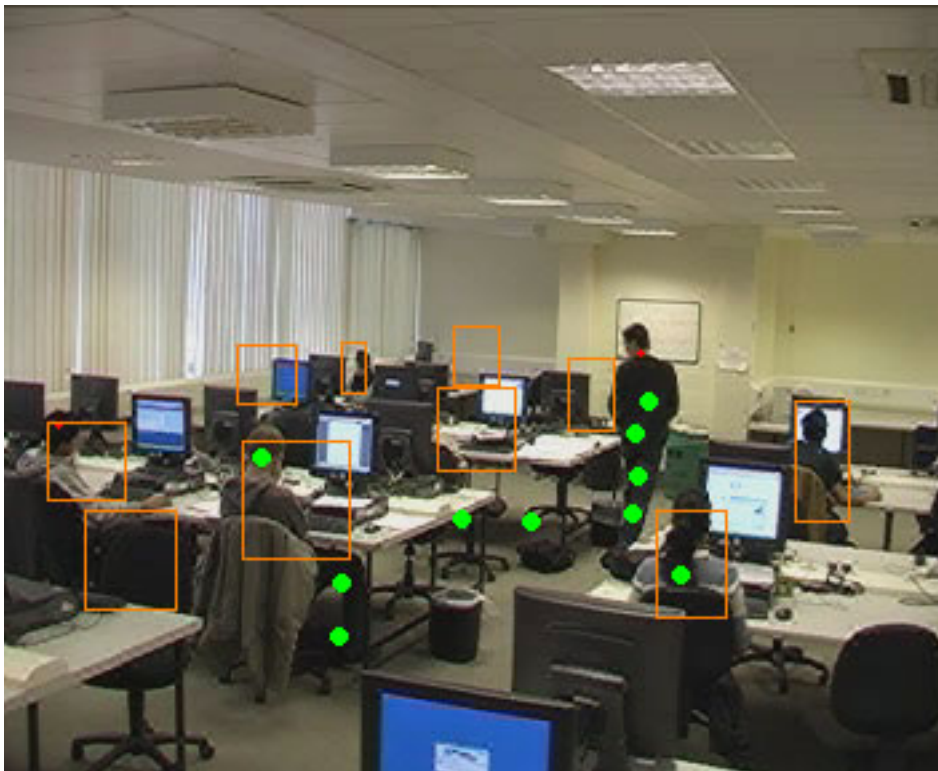
## **Appendix B: Colour Plates**

---

The following three pages contain colour plates A through E, please see chapter 3 for more details.



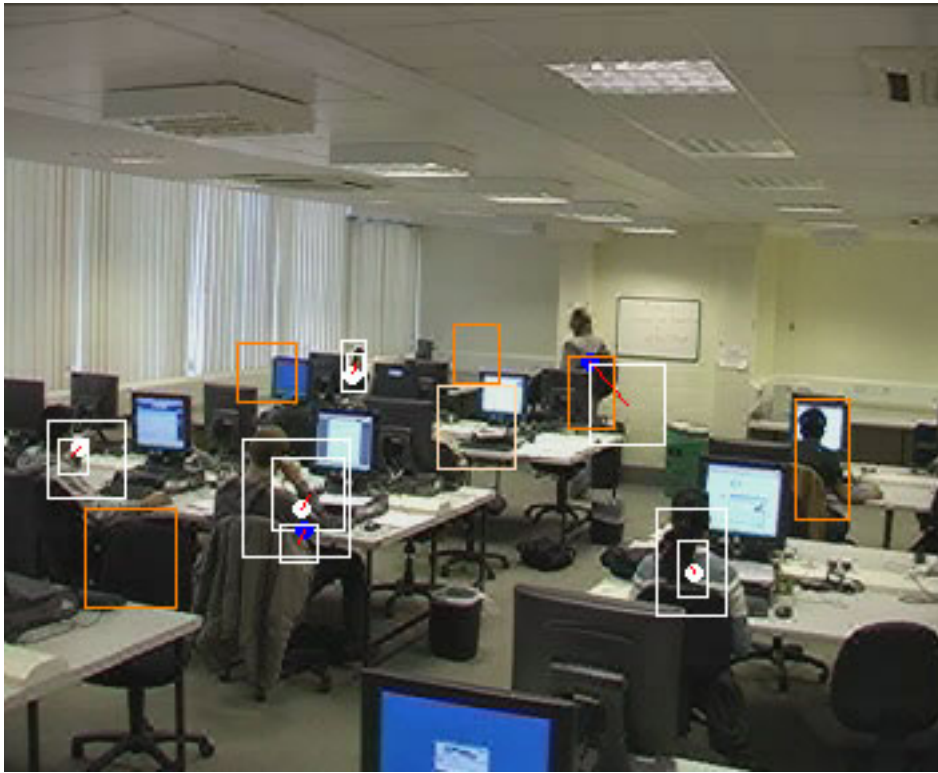
Colour Plate A: LUVG Output only



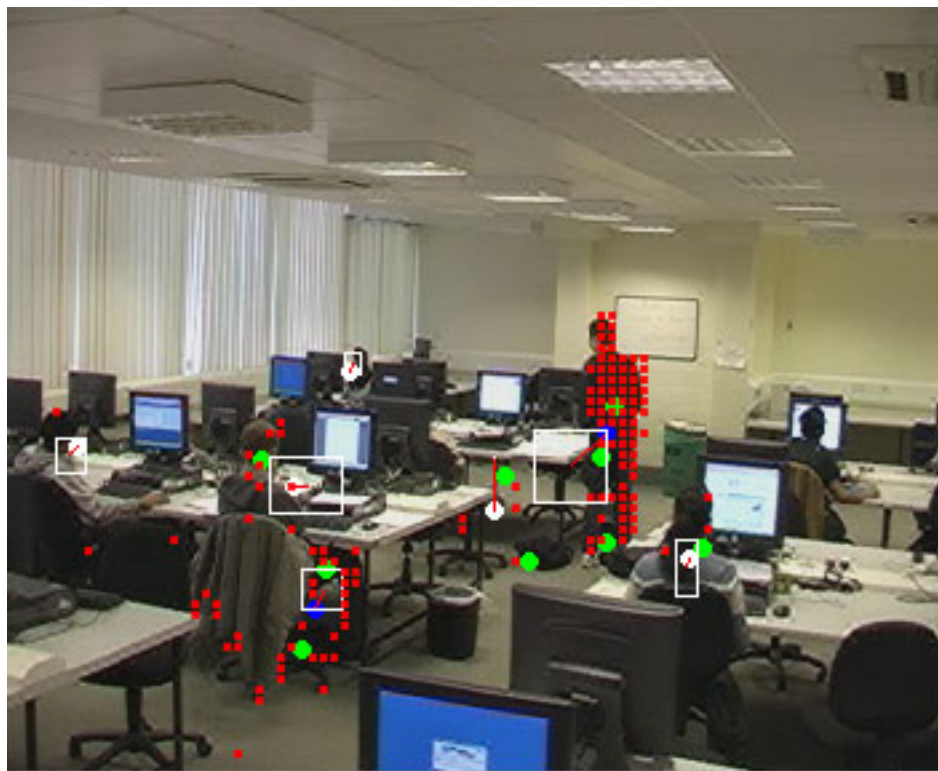
Colour Plate B: Example of FGM output and the stacked nature of free objects



Colour Plate C: Example of regioned and free object tracking. Notice as the student gets out of the chair, he moves it backward abruptly, and an additional free object is falsely assigned to it. Also note how, in the final frame, the student's bounding box passes through another regioned area without any ill effect



Colour Plate D: Example of weighted regions (note central terminal), and free-regioned occlusion



Colour Plate E (top to bottom): Foreground-background differentiation demonstration