# Project Proposal: Virtual Panning

Jared Norman, Chris Pocock, Terry Tsen
University of Cape Town

May, 2014

## 1 Project Description

The video recording of lectures has become increasingly popular with the advent of open courseware. Traditionally this has involved the use of a cameraman to film each lecture [Nagai (2009)]. The field of computer vision provides insight into automating this process, essentially removing the recurring costs of hiring someone to film each lecture [Wallick et al. (2004)].
Automation involves detecting features in the video and taking appropriate action. Either the detection is real-time, or it is done in post-processing. While some work has been done in the area of generating lecture videos automatically, these systems have various methods of solving the problem. In general, they have in common the following related components:

1. Camera setup and inputs
2. Feature detection and object tracking
3. Event detection and rules determining appropriate camera action

Our task is to create such a system. In this proposal, we will explain the reasons leading to this project, how we propose our system will work, and how we intend spending our time developing it.

## 2 Problem Statement

We have been approached by Stephen Marquard of the Centre for Innovation in Learning and Teaching (CILT, previously CET) with the task of creating a computer vision system capable of recording lectures with the following parameters:

1. The camera setup should be 3 static fixed HD cameras placed at the back of the lecture theatre.

2. The feature detection and object tracking should be able to track the lecturer.

3. A frame clipping of the stitched video stream should follow the lecturer as the lecture progresses.

The outcome of the project should be a system which produces video that is qualitatively comparable to a cameraman who pans with the lecturer as the lecture progresses.
One reason this project is worth doing is that most of the current solutions are either proprietary or require the use of an expensive Pan Tilt Zoom (PTZ) camera. These cameras cost around R60000, as opposed to the cost for 3 HD cameras which is around R4500. Such a system could be viable for anyone wishing to record lectures on the cheap which in turn supports open courseware initiatives and promotes mass education in general. The software we will use to develop the system is released under a BSD license and as such we are able to share our work at the source code level.
Our aim is to develop a cheap, automated recording system that allows for realistic lecture recordings without sacrificing image quality.

# 3 Procedures and Methods

We will be using the OpenCV library throughout the project. The project is divided into three distinct parts: *stitching*, *tracking* and *panning*. See Figure 1 for a visual representation of the work required for each of these components.

## 3.1 Stitching

This part of the project involves taking in three video streams and making one panoramic video stream. In this section, algorithms will be used to calculate the amount of overlap between each video stream. This is done by comparing feature elements of each video stream. During the process of stitching, visual artifacts along the stitching seams will need to be minimised. Some examples of artifacts include ghosting, vignetting and chromatic aberrations. The process of minimising visual artifacts is called blending. Blending involves manipulating pixel colours, such as averaging pixel colours within a certain area, to minimise stitch seams. The output of this stage is one panoramic video stream.

## 3.2 Tracking

The tracking component then locates the lecturer within the panoramic video stream created by the stitching component. This is done using face and/or body detection algorithms. These algorithms involve extracting features from the image and then determining if they form part of a face or body

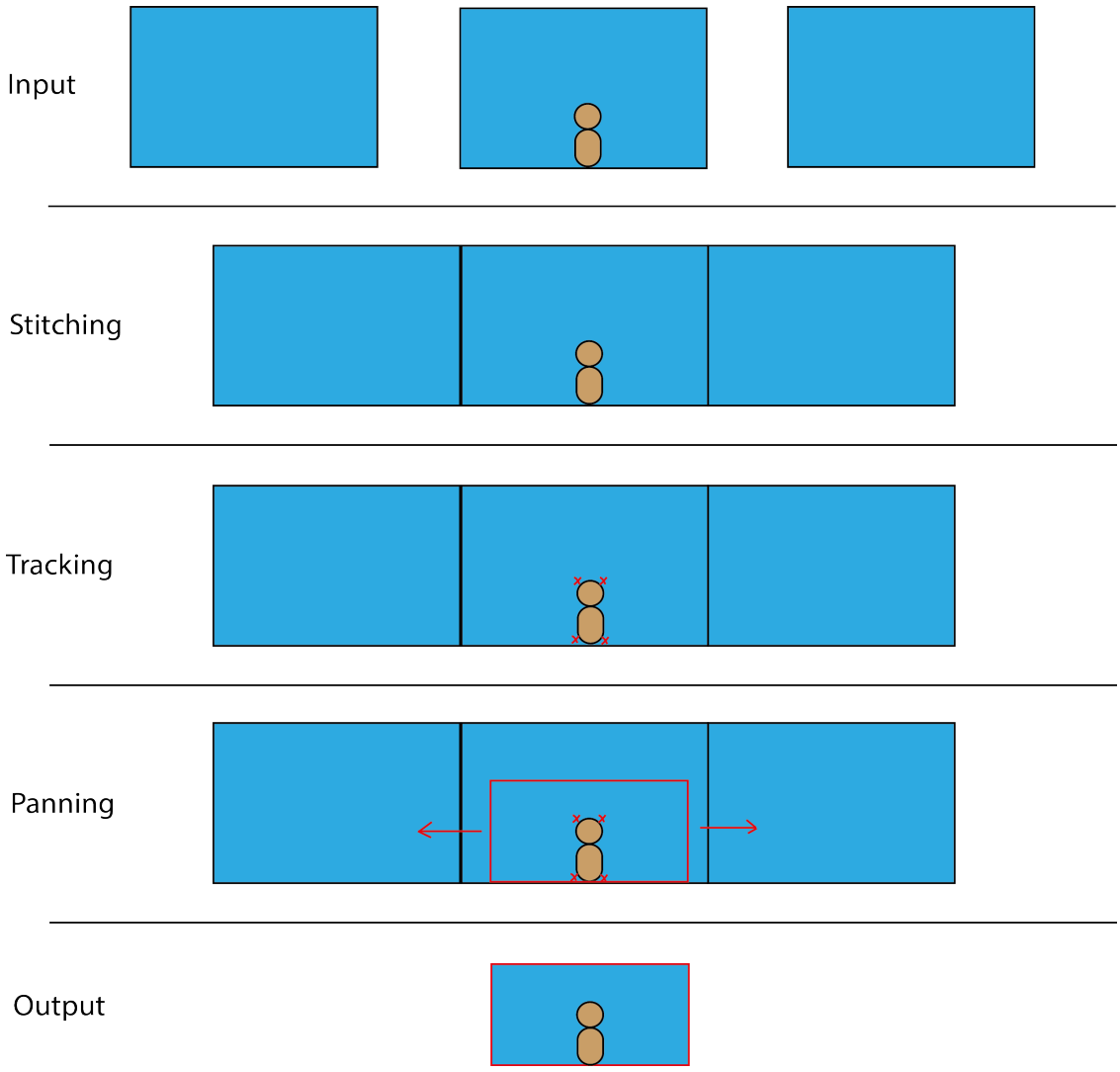


Figure 1: Illustration showing the distinct parts of the project, the inputs and the output

according to a classifier. Modern classifiers are trained using machine learning algorithms on large collections of example images. A good tracking algorithm should be robust to different lighting conditions, object size and orientation, clutter and occlusions. The output of this component is the position of the lecturer in the panoramic video stream.

## 3.3 Panning

Finally, the position of the lecturer as a discrete function of time, $p(t)$, is used in determining the position of the camera window as a function of time $c(t)$. It is not expected that the system be able to zoom in and out, so a fixed screen size will be agreed upon and the image will be clipped to that size. The panning component determines $c(t)$ according to situational analysis of the lecturer through $p(t)$. These notions can be mathematically framed in combination with heuristics taken from the field of videography. Rules will be determined as to what action the camera should take according to higher-level features of the video. These actions can also be mathematically framed as explained in Section 5.3

## 3.4 Evaluation of Individual Components

### 3.4.1 Stitching

Evaluation of the stitching component involves qualitatively describing the outputted panoramic video. The blending of the pixels must be to a degree that the stitch seams are non-existent in the output video. As such, the video must be watched without the viewer noticing any visual artifacts and/or distortions. The output video must give an illusion that the whole scene was taken with a single camera.

### 3.4.2 Tracking

The tracking component will be tested on a collection of video files. These videos will include the files created by the stitching component as well as individual video streams captured by one camera. In both cases, not all videos will contain faces or bodies (enabling the identification of false positives). During processing, the component will output a video file that includes red, rectangular regions of interest (ROI) that encapsulate a detected face or body (depending on which algorithm is being tested). These rectangles (Figure 2) provide a more visual representation of the component's performance than the numerical coordinates that are passed to the panning component. These output files will then be observed in order to qualitatively assess the accuracy of the component.

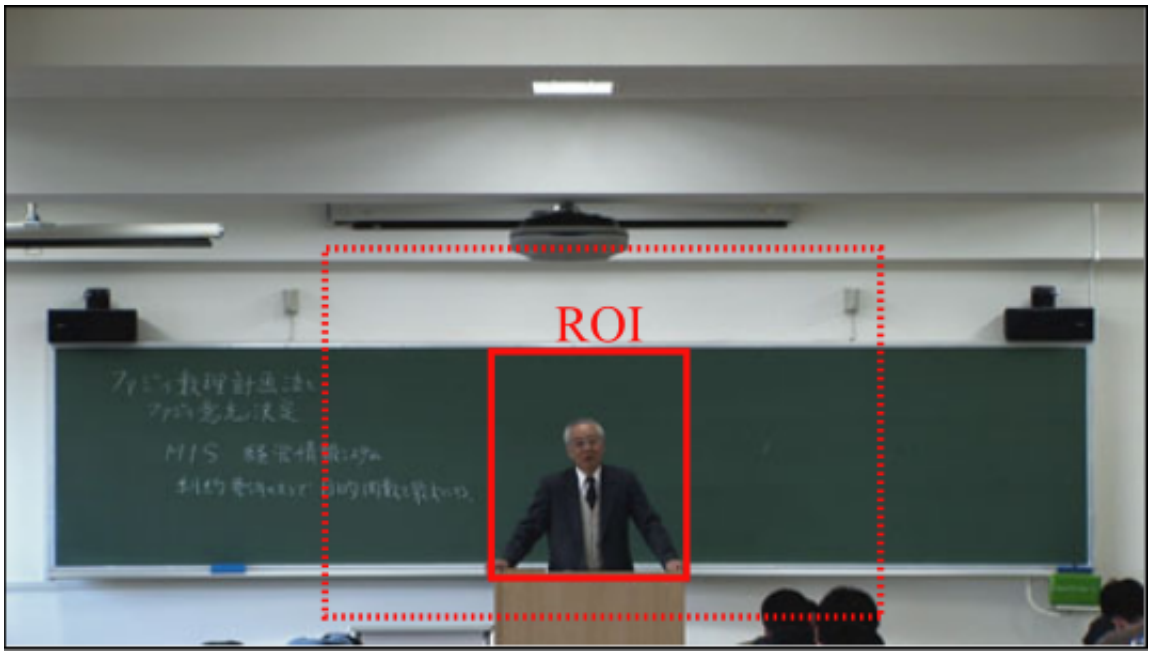


Figure 2: Region of interest and clipped frame found in [Yokoi and Fujiyoshi (2005)]

#### 3.4.3 Panning

Unit tests will be written to deal with functional requirements such as testing the virtual director (ie, given a constructed $p(t)$, can it determine the appropriate action) and testing the actual pan function (ie, given a location to pan to, does it pan according to the prescribed rules). Recognition of situations such as "lecturer moves quickly to the left" will require human verification undertaken by the developer by overlaying the relevant information (Figure 2) with captured video and empirically verifying that the results make sense.

### 3.5 Testing and Evaluation of System

The client will be the critic of whether the output video is of satisfactory quality. Development of the system will be deemed complete once both the client and our team are satisfied with it, or the final deadline is reached. More success factors can be found under Section 6.3

The development will be driven by unit tests for functional-level testing. These will test whether each part of each individual component works as intended (including input/output boundary testing). Integration of components should also be tested once components pass their unit tests. Once the system is deemed functionally complete, we will conduct informal testing. This involves making use of

the three-camera setup to be made available to us early in July. We may consequently change various parameters and restart the testing cycle. Once we feel the system conforms to the requirements, acceptance testing will be taken up with the client.

## 3.6 Design Features

OpenCV provides a wealth of GUI components in its HighGUI module. That said, our focus is on the functionality of the software with interface design being peripheral. It should be mentioned that OpenCV interfaces with Qt, a highly portable open source UI framework. We will leverage these as necessary.

## 3.7 Development Platform

We will develop the software using `libopencv-dev`, the OpenCV module for Ubuntu and similar Linux based systems. We are developing for Ubuntu because it was a requirement from the client. That said, the system is expected to be highly portable since it will be written in C++ (which is itself very portable). Further, we have found that it is difficult to set up OpenCV on Windows without using Microsoft's Visual Studio, which by default imposes a particular structure on the project files. For these reasons we believe Ubuntu to be a good choice of operating system on which to develop. We have chosen the OpenCV library due to its powerful range of tools, explicit documentation, helpful community and it being open source.

## 3.8 Implementation Strategy

We will follow the software engineering life cycle to implement the system.

1. **Requirements gathering/analysis**: This stage of the life cycle involves asking the client what is expected of the system. The client gives us a list of constraints and requirements which we must adhere to when developing the system.

2. **Software architecture**: After the requirements are gathered, we need to come up with an architecture of the system. This has been provided by our supervisor, Dr. Patrick Marais, namely that we should use OpenCV as a base for our system.

3. **Computer programming**: This stage involves programming the system.

4. **Testing**: This stage involves testing the system to determine not only that it works, but also that it conforms to the client's constraints and requirements. Since we will be using an agile approach, stages 3 and 4 will interchange over the course of development of the system.

5. **Quality assurance**: The client will determine whether the system is what they expected.

### 3.9 Expected Challenges

One of the main challenges we expect to encounter will be the qualitative testing of individual components and the overall system. Assessing performance over an entire lecture recording will be a timely, unavoidable task.
Some other challenges are anticipated due to the use of the agile approach. The agile approach requires a lot of meetings and collaboration with the client, which can be time-demanding and interrupt the development process. Additionally, the iterative nature of agile development means that testing (an already lengthy task) will have to occur repeatedly.

## 4 Issues

There are some issues that we have foreseen at the beginning of the project and they are discussed below.

### 4.1 Ethical

With regards to getting input data, we may need to request permission from the lecturer that the lecture recording will be used for experimental purposes. We do not plan on performing any user experiments, since the usability and completion of the system will be determined by the client on behalf of the CILT.

### 4.2 Professional

We should note that while we wish to make the system as general as possible, the development is guided toward the approval of the client and should be conducted in a professional manner. Thus any additional features we might want to include first need to be approved by the client. We also do not foresee any personal injury or harm from use of the system.

### 4.3 Legal

The library we will be using, OpenCV, is free for use under the open source BSD license. Since the project might become a basis for future projects in this field, we will release the work under an open source license provided UCT allows this (since technically UCT hold a copyright over any Honours projects).

# 5 Related Work

While some systems employ PTZ cameras [Chou et al. (2010)],[Lampi et al. (2007)],[Wallick et al. (2004)],[Winkler et al. (2012)], and other technologies such as Microsoft's Kinect device [Winkler et al. (2012)], which enables depth tracking, we are more concerned with work employing static cameras such as those found in [Ariki et al. (2006)],[Gleicher and Masanz (2000)],[Nagai (2009)],[Kumano et al. (2005)], and [Yokoi and Fujiyoshi (2005)].
Related works of each part of the project will be discussed in detail below.

## 5.1 Stitching

In order to find the amount of overlap between pairs of video streams, feature points need to be identified. These feature points are also known as "interest points". To make identification of interest points as robust as possible, Lowe's Scale Invariant Feature Transform (SIFT) will be used [Brown and Lowe (2003)]. This means that the detected features are robust to image scale and rotation, distortion, change in 3D viewpoint, noise and change in illumination [Szeliski (2006)]. Once the amount of overlap is determined and the video streams stitched together, pixel colours along the stitch seams will be blended together. [Triggs et al. (2000)] describes two methods: the pixels in an area can be averaged to form a final pixel, or a weighted average can be used (pixels in the centre are weighted more heavily than the pixels on the edges).

## 5.2 Tracking

As has been mentioned, most tracking algorithms involve detecting objects by extracting features from an image and then determining if they form part of a greater object according to some type of classifier. Of the thousands of features that are detected, most are irrelevant (not part of the greater object). A method proposed by Viola and Jones uses a machine learning algorithm called AdaBoost (Adaptive Boost) to select only the most relevant features for classification [Viola and Jones (2001)]. The selected features are then classified by cascade of small, efficient classifiers in stages. This combination of AdaBoost and classifier cascade rapidly increases the algorithm's speed. This is the method employed in the lecturer recording system found in [Chou et al. (2010)]. Another method, called HOG (Histograms of Oriented Gradients), divides the image into cells and, for each cell, calculates a histogram of gradient directions of the cell's pixels [Dalal and Triggs (2005)]. These histograms combined describe the image and are classified by a Support Vector Machine.

### 5.3 Panning

Not surprisingly, this component has been a part of most of the previous systems. Where a PTZ camera was used, this component would select new co-ordinates and change the image plane accordingly (as the camera pans). Here, we are not concerned with PTZ cameras and so the situation is simpler. However, we are able to look forward in time. The system presented in [Chou et al. (2010)] makes use of a camera action table which takes a recognized state (such as "Status of Lecturer = Toward to the screen and Face = Face to screen") and determines a camera action (such as "The zoomed-out image contains part of the screen"). Such an idea is also encapsulated with Finite State Machines in [Lampi et al. (2007)] and [Wallick et al. (2004)] and with a scriptable virtual director in [Wulff and Fecke (2012)]. The higher-level situations discussed above are determined using the results of the object tracking. For instance, [Chou et al. (2010)] tracked the lecturers face and the board, while [Lampi et al. (2007)] make use of skin tone, edge and motion detection.
Lastly, [Yokoi and Fujiyoshi (2005)] designed a system making use of temporal differencing and bilateral filtering to extract a smoothly moving centroid (geometric centre) of the lecturer. An algorithm was developed to determine how and when to pan. Videographic heuristics found in the literature were used. For example, the fact that most panning between points would decelerate 60% of the total panning time and that the panning speed reaches its maximum around the transition point from acceleration to deceleration, they were able to construct an interpolation function of the frame co-ordinates which mimics professional cameramen. Such a function corresponds to a part of $c(t)$ mentioned in Section 3.3. In general we consider it a piece-wise defined, discrete function.

## 6 Anticipated Outcomes

### 6.1 System

We expect to produce a software artifact that is capable of receiving 3 video feeds, stitching them together to produce 1 panoramic video feed, tracking the presenter and cropping the video that will pan along with the presenter.
The software artifact should run on Ubuntu. It should not utilize GPUs to accelerate its performance. Lastly, the post-processing time for the video should be no longer than 3 times the run time of the video. Essentially the client will be able to install the system on a Linux distribution with minimal fuss and make use of it in the recording of lectures.

## 6.2 Expected Impact

Stitching, tracking and panning have been around since the 1980's. Automatic lecture recording systems have been developed at least since the 2000's. That said, no automatic lecture recording system has been created in the proposed manner, which enables people on a small budget to record lectures. It is for this reason that we plan to release our project under a BSD license at the conclusion of the project

## 6.3 Key Success Factors

The main success factor of the project is whether or not the client is satisfied with the output video quality of our system. CILT plans on rolling out the virtual panning system to some lecture theatres in UCT; if this happens, we will consider our project to be highly successful. The project will also be successful if CILT continues to develop the system well into the future to, for example, incorporate lecture slides.

# 7 Project Plan

## 7.1 Risks

### 7.1.1 Delayed acquisition of video data

As has been mentioned, we will need three-camera lecture video recordings in order to develop and test our system properly. If this is delayed, our project may fall behind schedule.
*Likelihood*: Low
*Severity*: Medium
*Mitigation strategy*: We can use three webcams and our own venue to record video data. This will allow us to continue development and stay on schedule. When we eventually receive the recordings from CILT, we will adapt our system accordingly.

### 7.1.2 Loss of work

As this is a long term project with lots of code and files, the chance of losing work is increased. Depending on the amount of work lost, this could delay or even destroy the project. In either case, developers would need to backtrack and start again.
*Likelihood*: Medium
*Severity*: Medium to High
*Mitigation strategy*: All code will be synchronised with the Git version control service. Documentation

will be stored on cloud services like Google Drive. As an added precaution, code will be backed up onto external storage when development milestones are reached.

#### 7.1.3 OpenCV inadequate

As this is the first time the developers are working with the OpenCV library, we are unsure of its capabilities. If OpenCV is unable to support our main components sufficiently, the performance of our system may be unsatisfactory. However, related work that uses OpenCV indicates that it should meet our requirements.
*Likelihood*: Low
*Severity*: Medium
*Mitigation strategy*: OpenCV is extensible, so we can attempt to add needed functionality that isn't there already. Alternatively, we could decide on a different computer vision library to use. While many of these libraries are proprietary, there are other open source alternatives such as SimpleCV and VLFeat.

#### 7.1.4 Failure to integrate three main components

The stitching-tracking and tracking-panning integrations are vital. If all three components work independently but one of these links is insufficient or nonexistent, the overall system will not work.
*Likelihood*: Low to Medium
*Severity*: High
*Mitigation strategy*: Before developing the components independently, the team will agree on the interfaces between their components. We will maintain and test these interfaces between development cycles. This planning and ongoing coordination should make the integration phase smoother and quicker.

#### 7.1.5 Failure to deliver on time

It is expected that the deliverables for the project will be very time-consuming. If the team or a member falls behind, it may be unrealistic for them to catch up and meet deadlines. This is very undesirable as it could result in a rushed or incomplete system.
*Likelihood*: Medium
*Severity*: Medium
*Mitigation strategy*: The team will meet regularly with each other and their supervisor to ensure that they are on schedule and not at risk of missing deadlines. The team will also use agreed-upon

communication channels to raise issues early. Additionally, the team will follow the project timeline (shown in the section below).

## 7.2 Required Resources

### 7.2.1 Software and Data

- Lecture video recordings from 3 different camera feeds
- OpenCV library
- C++ editor
- Linux OS
- Git repository (BitBucket)

### 7.2.2 Hardware and Equipment

- 3 development workstations
- 3 HD IP cameras
- Access to test venues

### 7.2.3 People

- Supervisor: Dr. Patrick Marais with regular guidance and input from Stephen Marquard of the CILT.
- Development team: Jared Norman, Chris Pocock, Terry Tsen

## 7.3 Deliverables

The project includes seven main deliverables. Table 1 shows each one of these deliverables and their due date.

## 7.4 Timeline

The Gantt chart (found in Figure 3) represents the timeline of our project. The different categories of work are represented by five different colours; purple (planning), yellow (website), green (development), red (report) and blue (other).

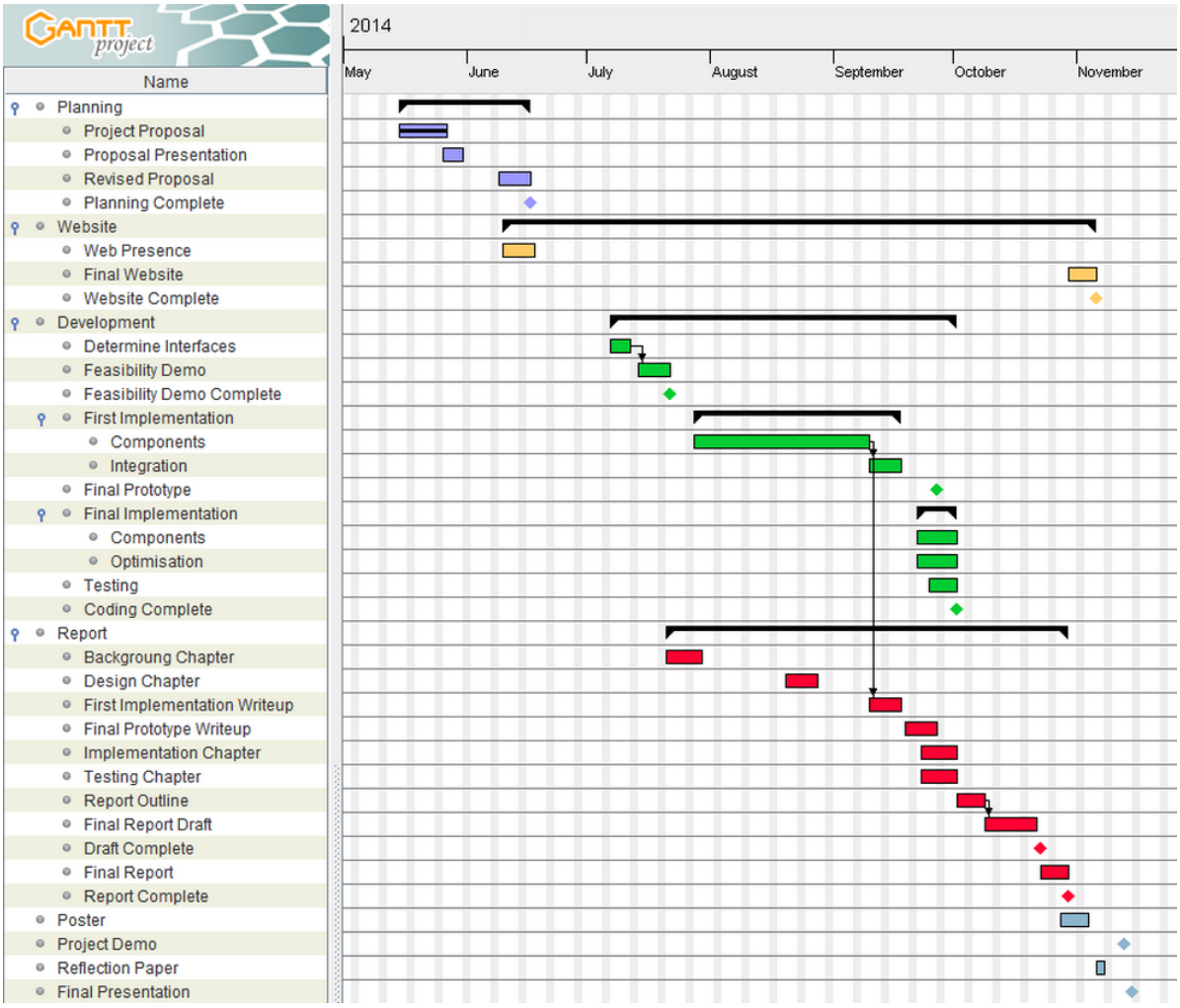


Figure 3: Gantt chart showing progression of the project and submission dates

| Deliverable | Date |
|---|---|
| Literature Reviews | 15th May |
| Research Proposal | 26th May |
| Initial Feasibility Demo | June/July |
| Project Report | 29th September |
| Poster | 3rd November |
| Web Page | 5th November |
| Reflection Paper | 9th November |

Table 1: Deliverables and due dates

## 7.5 Milestones

The major milestones for the project and the dates by which they are expected to have been completed can be found in Table 2. These milestones are also represented by the diamonds in the Gantt chart.

| Milestone | Date |
|---|---|
| Planning Complete | 16th June |
| Feasibility Demo | 21st July |
| Final Prototype | 26th September |
| Coding Complete | 1st October |
| Report Draft | 22nd October |
| Report Complete | 29th October |
| Website Complete | 5th November |
| Project Demo | By 12th November |
| Final Presentation | 14th November |

Table 2: Project milestones and due dates

## 7.6 Division of Work

The proposed system has been divided into three main components that require roughly the same amount of work to research and develop. After agreeing upon interfaces between these components, they can each be worked on independently.

*Terry Tsen* will work on the stitching component where he will synthesize a large virtual image from three separate camera feeds. Within this virtual image, *Chris Pocock* will detect the lecturer over multiple frames. Lastly, *Jared Norman* will use the virtual image and the position of the lecturer to

create a cropped, lecturer-centered video stream.

## References


Yasuo Ariki, Shintaro Kubota, and Masahito Kumano. 2006. Automatic production system of soccer sports video by digital camera work based on situation recognition. In *Multimedia, 2006. ISM'06. Eighth IEEE International Symposium on*. IEEE, 851–860.

Matthew Brown and David G. Lowe. 2003. Recognising panoramas.. In *ICCV*, Vol. 3. 1218.

Han-Ping Chou, Jung-Ming Wang, Chiou-Shann Fuh, Shih-Chi Lin, and Sei-Wang Chen. 2010. Automated lecture recording system. In *System Science and Engineering (ICSSE), 2010 International Conference on*. IEEE, 167–172.

Navneet Dalal and Bill Triggs. 2005. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, Vol. 1. IEEE, 886–893.

Michael Gleicher and James Masanz. 2000. Towards virtual videography (poster session). In *Proceedings of the eighth ACM international conference on Multimedia*. ACM, 375–378.

Masahito Kumano, Yasuo Ariki, and Kiyoshi Tsukada. 2005. *A method of digital camera work focused on players and a ball*. Springer, 466–473.

Fleming Lampi, Stephan Kopf, Manuel Benz, and Wolfgang Effelsberg. 2007. An automatic cameraman in a lecture recording system. In *Proceedings of the international workshop on Educational multimedia and multimedia education*. ACM, 11–18.

Takayuki Nagai. 2009. Automated lecture recording system with AVCHD camcorder and microserver. In *Proceedings of the 37th annual ACM SIGUCCS fall conference*. ACM, 47–54.

Richard Szeliski. 2006. Image alignment and stitching: A tutorial. *Foundations and Trends® in Computer Graphics and Vision* 2, 1 (2006), 1–104.

Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. 2000. *Bundle adjustment—a modern synthesis*. Springer, 298–372.

Paul Viola and Michael Jones. 2001. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, Vol. 1. IEEE, I–511–I–518 vol. 1.

Michael N. Wallick, Yong Rui, and Liwei He. 2004. A portable solution for automatic lecture room camera management. In *Multimedia and Expo, 2004. ICME'04. 2004 IEEE International Conference on*, Vol. 2. IEEE, 987–990.

Michael Bjorn Winkler, Kai Michael Hover, Aristotelis Hadjakos, and Max Muhlhauser. 2012. Automatic camera control for tracking a presenter during a talk. In *Multimedia (ISM), 2012 IEEE International Symposium on*. IEEE, 471–476.

Benjamin Wulff and Alexander Fecke. 2012. LectureSight-An Open Source System for Automatic Camera Control in Lecture Recordings. In *Multimedia (ISM), 2012 IEEE International Symposium on*. IEEE, 461–466.

Takao Yokoi and Hironobu Fujiyoshi. 2005. Virtual camerawork for generating lecture video from high resolution images. In *Multimedia and Expo, 2005. ICME 2005. IEEE International Conference on*. IEEE, 4 pp.