project AVRiL

Final Report
Version 0.1

Advisors:
Dr. Sohaib Khan
Dr. Umar Saif

BSc Sproj '07 - Group 7:
Ahmad Humayun
Ozair Muazzam
Tayyab Javed
Yahya Cheema

avril.sproj.com
STATEMENT OF SUBMISSION

A report submitted to the Department of Computer Science and the Department of Computer Engineering in partial fulfillment of the requirements for the degree of BSc (Honors) in Computer Science and BSc (Honors) in Computer Engineering by

Ahmad Humayun for BSc (Honors) in Computer Engineering
Ozair Muazzam for BSc (Honors) in Computer Science
Tayyab Javed for BSc (Honors) in Computer Science
Yahya A. Cheema for BSc (Honors) in Computer Science

Advisors

Dr. Sohaib Khan Department of Computer Science
Dr. Umar Saif Department of Computer Science

Lahore University of Management Sciences

July 1st, 2007
ACKNOWLEDGEMENTS

Above all we would like to thank Allah. Without His blessings this project wouldn’t have even lifted off the ground. We would like to thank Dr. Sohaib Khan for guiding us throughout the project, and putting us back on track whenever we have wandered away from it. He has remained a true inspiration to us all. We also appreciate Dr. Umar Saif for motivating us over the last year, and showing infinite patience to our requests at awkward hours.

We would also like to thank the Computer Science Department at LUMS. Dr. Shafay Shamail, the Head of the Department, has offered us help where ever he could, and made the development logistically simpler.

Finally, we would like to thanks our families, who have prayed for us throughout, every time we have succeeded or fallen in trouble. They have been extremely understanding whenever we have had to stay in our labs for that extra hour, in thinking and developing our next innovation.

Ahmad Humayun  2007-02-0236
Ozair Muazzam  2007-02-0162
Tayyab Javed       2007-02-0130
Yahya Aman Cheema  2007-02-0370

Dated,
July 1st, ‘07
# TABLE OF CONTENTS

STATEMENT OF SUBMISSION ................................................................. ii
ACKNOWLEDGEMENTS ........................................................................ iii
TABLE OF FIGURES ........................................................................... viii
ANALYSIS ............................................................................................. 1
  Problem Statement ........................................................................... 2
  Current Systems ............................................................................. 3
    AutoAuditorium - Bianchi ‘99 ......................................................... 3
    Microsoft Research (MSR) iCam and iCam2 – Yong R, et al., ’01, ’03, ’05 .. 3
    CARMUL - Kyoto University – Kameda Y., et al. ‘03 ..................... 4
    Cornell Lecturer Browser – Mukhopadhyay S., Smith B., ‘99 .......... 5
Possible Solutions ............................................................................. 6
  Dais location ................................................................................... 6
  Lecturer detection and tracking ...................................................... 6
  Board Contents ............................................................................. 6
  Slides ............................................................................................ 7
  General Audience Location ......................................................... 7
  Active Audience Location ............................................................ 7
Proposed System ............................................................................. 8
  Dais location ................................................................................... 8
  Lecturer detection and tracking ...................................................... 8
  Board Contents ............................................................................. 8
  Slides ............................................................................................ 8
  General Audience Location ......................................................... 8
  Active Audience Location ............................................................ 8
DESIGN .............................................................................................. 9
  DFD Level 0 .................................................................................. 10
  DFD Level 1 .................................................................................. 11
IMPLEMENTATION ........................................................................... 12
  System Components ...................................................................... 13
    Hardware Level .......................................................................... 13
      Cameras .................................................................................. 13
      Lecturer Tracking Camera ..................................................... 13
      Audience Tracking Camera ................................................... 13
    Microphone Array .................................................................... 14
  Raw Data Level .......................................................................... 14
    Video-Receiver ......................................................................... 14
    Camera Controller .................................................................... 14
    Audio Processing ..................................................................... 15
  Software Tracking Level .............................................................. 15
    Slide Transition Detection (also part of Raw Data Level) ............ 15
    Physical Design Rules .............................................................. 15
    Lecturer Tracking ................................................................. 15
EXECUTIVE SUMMARY

We think that the basic problem faced by education today is not the absence of quality education. Many universities have the right content to educate everyone, but of course recording lectures through traditional means requires a lot of resources. The problem with education is the need to make quality educational content creation cheaper. This will allow more people to obtain access to better education.

The aim of the project is to design a system that would automate the recording process for university lectures, using video cameras that act intelligently to focus on the instructor, the presentation, the dais or the class of students, whichever is more important at that time.

There is a dire need of cheap, deployable, video systems which are able to record an academic atmosphere un-aided. We aim to develop a system which doesn’t require human intervention because of two reasons; employing a dedicated camera team doesn’t allow a university to film every important lecture feasibly; two, the visual presence of someone guiding a camera might not allow an instructor to follow his own natural style of teaching.

AVRiL (Automated Video Recording of Lectures) aims to automatically record, using PTZ cameras, university lectures in order to assist distance learning programs. We are assuming that the system would be used inside auditoriums only, in which only one instructor is teaching at a time. We also assume that the instructor will be standing and in motion most of the time, teaching a group of seated students who face away from the camera. Our final aim is to produce a high quality presentation video which retains a feel of the classroom feel.
# TABLE OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Data Flow Diagram Level 0</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>Data Flow Diagram Level 1</td>
<td>11</td>
</tr>
<tr>
<td>3</td>
<td>Camera Panning in action</td>
<td>17</td>
</tr>
<tr>
<td>4</td>
<td>Different shapes of Auditoriums</td>
<td>18</td>
</tr>
<tr>
<td>5</td>
<td>Elmo camera used in AVRiL</td>
<td>19</td>
</tr>
<tr>
<td>6</td>
<td>Frame zooming</td>
<td>20</td>
</tr>
<tr>
<td>7</td>
<td>Two frames showing the large motion in audience</td>
<td>25</td>
</tr>
<tr>
<td>8</td>
<td>Motion through animation on slides. Notice the line appearing on the lower left of the slide</td>
<td>26</td>
</tr>
<tr>
<td>9</td>
<td>Camera movement during a lecture</td>
<td>26</td>
</tr>
<tr>
<td>10</td>
<td>Difference in visibility from change lighting conditions</td>
<td>27</td>
</tr>
<tr>
<td>11</td>
<td>Change in board content during the lecture (Courtesy Massachusetts Institute of Technology ©)</td>
<td>27</td>
</tr>
<tr>
<td>12</td>
<td>Contrast the distinctness of the lecturer from the background in the two frames</td>
<td>28</td>
</tr>
<tr>
<td>13</td>
<td>Background Subtraction run on a background image and a frame taken during a lecture</td>
<td>29</td>
</tr>
<tr>
<td>14</td>
<td>Frame differencing operation illustrated</td>
<td>30</td>
</tr>
<tr>
<td>15</td>
<td>Shape of a 2D Gaussian for taking weighted average of pixels</td>
<td>31</td>
</tr>
<tr>
<td>16</td>
<td>Steps of Lecture Detections</td>
<td>32</td>
</tr>
<tr>
<td>17</td>
<td>Initial Board writing legibility tests</td>
<td>47</td>
</tr>
<tr>
<td>18</td>
<td>Haar-like features used by the current OpenCV algorithm</td>
<td>51</td>
</tr>
<tr>
<td>19</td>
<td>The Integral image trick. After integrating the pixel at (x,y) contains the sum of all pixel values in the shaded rectangle. The sum of pixel values in rectangle D is (x4, y4) - (x2, Y2) - (x3,y3) + (x1, y1).</td>
<td>52</td>
</tr>
<tr>
<td>20</td>
<td>The classifier cascade is a chain of single-feature filters. Image subregions that make it through the entire cascade are classified as &quot;Face.&quot; All others are classified as &quot;Not Face.&quot;</td>
<td>53</td>
</tr>
<tr>
<td>21</td>
<td>The first two Haar features in the original Viola-Jones cascade</td>
<td>53</td>
</tr>
<tr>
<td>22</td>
<td>MeanShift at work on face tracking</td>
<td>54</td>
</tr>
<tr>
<td>23</td>
<td>Results sequence shown in the original MeanShift paper</td>
<td>56</td>
</tr>
</tbody>
</table>
CHAPTER 1

ANALYSIS
Problem Statement

AVRiL aims to revolutionize the access to education for the less fortunate, while facilitating the ones already receiving high standards of education. The goal is to completely automate the classroom lecture-recording process, eliminating all human elements including the cameramen and the director, thereby reducing cost and yet producing unobtrusive, high-quality video with an immersive "classroom look-and-feel". This will make educational content-creation accessible and affordable for all universities and colleges to record day-to-day lectures and publish them on the Internet.

Quality education is a scarce resource in this world. Out of the thousands of schools and universities, only a select few offer high-quality education and most of them are located in the developed world. Therefore, a large population of the world is still deprived of this highly valued pre-requisite to a progressive world. Modern telecommunication technologies, and in particular the Internet, have created a platform that enables ease in information sharing. There have been numerous attempts by schools and universities at leveraging these technologies to share quality education with those who do not have access to it. However, most of these attempts have been unsuccessful because of a lack of the “classroom look-and-feel.” We believe that this is essential for a student viewer to continuously stay interested throughout a lecture as well as the entire course. The next best thing to being in class is to give the student this feel by providing him/her with almost all the elements of an actual class. The most effective way to achieve this would be to create an interesting video of the lecture, encompassing the lecturer, the audience, the presentation slides and the blackboards.

The traditional approach to recording lectures involves the hiring of a crew, which includes several cameramen and a director (apart from other support staff). The estimated cost of such a recording ranges between USD 300 and USD 2000. For a university offering 200 courses, recording course lectures once would cost in millions. This is why we feel that today the real gap in spreading education using technology is not in content distribution, but rather in quality content creation at an affordable price.

AVRiL achieves this goal by using computer vision techniques to understand the lecture environment and keep track of the lecturer and the students. Further, sound localization techniques are used to track student participation in the class. These heuristics combined with cinematography rules make an interesting and captivating video. This video is made available using an intuitive, bandwidth-sensitive web interface for students to enable them to experience the classroom environment as naturally as possible.
Current Systems

During the last few years, especially since the advent of broadband internet, there have been quiet a few attempts at automatic recording of presentations, podium talks and even classroom lectures. We did a thorough survey of all publicly available research pertaining to such systems. Here is a list of a few ventures that were quiet similar to what we wanted-

**AutoAuditorium - Bianchi ‘99**

This is a very popular commercial system that uses proprietary computer vision techniques to keep track of the lecturer and also includes direction capability for automatic shot selection.

The system has been sold to leading places like IBM Research and MIT and has been used to record hundreds of lectures and presentations every year.

As compared to recent much more advanced systems, AutoAuditorium can do audio-mixing and amplify the appropriate voice signal when the audience is speaking, but it does not have localization capability

-Features:
  - Automatic video capture of real-world presentation using **speaker tracking**.
  - Automatic shot selection, **video mixing** using built in **direction rules**.
  - Audio mixing by selecting appropriate audio signal to differentiate speaker voice from audience participation.

**Microsoft Research (MSR) iCam and iCam2 – Yong R, et al., ’01, ’03, ’05**

This system has been the pioneer over the last few years which has lead research in this area to new heights. Over the last 6 years their team has published 3 papers, 2 of them describing full systems and one describing videography rules for recording lectures. A number of people have worked in MSR for the development of this system, out of which the most significant contributor has been Yong Rui, and lately Cha Zhang.

Over the years they have developed successfully systems which can completely automate the lecture recording process. The iCam and the iCam2 system have not only recorded 540+ lectures but also broadcasted them over the network.

Salient points of the 3 papers
  - **iCam** used four analog cameras: a speaker tracking camera, an audience tracking camera, a static wide-angle camera and a scan convertor camera for slides. The
iCam2 system is a much more simplified implementation having one digital lecturer camera and one audience camera, both connected to the Ethernet.

- iCam uses frame differencing and motion detection for lecturer tracking whereas iCam2 employs a more sophisticated method tracking using motion histograms (refer to Motion Tracking on page 28).
- iCam uses an array microphone to pan-tilt-zoom the audience camera to capture audience participation.
- iCam2 lecturer tracking camera replaces three cameras of iCam by not only tracking the lecturer but also mimicking the work of the wide-angle camera and the scan convertor camera. It acts as the slide camera whenever there is motion / animation on slides.
- iCam introduced the Virtual Video Director module which implements cinematography rules through a finite state machine structure. It runs by taking by looking at the status vector of each camera (ready / not ready / general) and shot lengths. iCam2 built on top of it by looking at lecturer motion to get cues for pan-tilt-zoom levels.
- In the Videography for Telepresentations paper, MSR proposed direction rules in more detail through a Virtual Director (VD) and Virtual Cameramen (VC) and defining rules for lecturer tracking, audience tracking, camera positing and shot transitioning.

**CARMUL - Kyoto University – Kameda Y., et al. ‘03**

This system has been tried and tested at the Kyoto university and has been regularly used to record regular lectures on the campus. Since October 2002 they have conducting live tests to archive their lectures by recording regular classes of courses running 6 times a week for 15 weeks.

CARMUL also targets audiences for distance learning. They have developed a web-module for their system based on RealNetworks which shows synchronized content of a given lecture. Their system introduced many innovative ideas by compromising on unobtrusiveness requirements for such a system.

Features:

- **Obtrusive means to track lecturer** using ultra-sonic beacons.
- **Uses lecturer trajectory** for cues to detect the status of the lecture.
- **Microsoft PowerPoint plug-in to note slide transitions.**
- **Use of electronic whiteboard to record hand-writing** on board. The synchronization is done by recording time-stamps every time the lecturer writes something.
- **Hybrid location estimation** using acoustic and vision analysis to detect speaking student.
- **Direction totally based on cues from the tracking devices.**
Cornell Lecturer Browser – Mukhopadhyay S., Smith B., ‘99

The most interesting part in this research has been the methods taken to track slide transitioning without the use of powerpoint plugins. Using the overhead camera, the system tracks slide transition times and segments the whole video according to slides by using adaptive thresholding and dilation to match slides on different frames of the video.

Main Contributions:
• Synchronization of timed and un-timed data.
• Automatic synchronization of slides without any software plug-ins. By using a separate overhead camera for slides, the system tries to match slides to video frames.
• Automatic editing through heuristic algorithms to make Edit Decision Lists (EDL).
• Direction based on shot-time elapsed and slide transitions.
Possible Solutions

From our research we found that there were several ways of tackling the problem. One fundamental part was to make the system aware of the changes in the environment so it could adapt accordingly. The bits of important information needed from the environment are-

1) The location of the dais
2) The location of the lecturer
3) The board contents
4) The slides
5) General Audience location
6) Active Audience location

Dais location

We decided that it was best to have a person identify the dais before a recording starts rather than having the camera automatically locate it. The possible means for automatic location all involved the use of some artificial marker placed on the dais but that was unnecessary since the camera location could be made such that a human can initially point it to the dais.

Lecturer detection and tracking

For detection we could similarly have a person point out the lecturer at the start of a recording or we could use an automatic means both with markers – using RF beacons as used by Kyoto University, or special color markers that we could identify using the camera. We also had the option of using simpler methods such as background subtraction or advanced vision/AI techniques for human detection such as a trained haar detector.

For tracking we had the option of similarly iterating on our chosen detection method, such as background subtraction, markers, RF beacons, human detection etc… or using other methods based on the current detection output that we have gotten-such as motion detection and color detection further augmented with heuristics on possible lecturer movement.

Board Contents

For board detection we had the option of having a dedicated camera capturing the relevant portion of the board at any particular time; or we could use a single camera and take periodic shots of the board. For really good results we also had the option of using electronic white-boards as used in the Classroom 2000 project.
**Slides**

For slides capturing, we could use a camera to record the slides as they are presented; have a plug-in for the slide presentation software that exports the slides and the presentation timestamps; or use a built-in screen capture mechanism to ensure that we can capture everything happening on the screen including demos of software.

**General Audience Location**

Periodically we also need to capture shots of the audience participating in a class and it is important for the system to know their location. For this, we can similarly have an area of the audience marked by our Mr. Initial Marker but we still need more information before we zoom so that we know that we’re not zooming into empty seats. This can be done using a background subtraction technique or by using a face detector.

**Active Audience Location**

Not knowing where an audience is sitting is not enough, we also need information on the location of ‘active’ audience, such as those asking a question during the lecture. For this we need sound information and where it is coming from. One way to do this is to use a microphone array for sound-source localization.
Proposed System

Most of the decisions that we made on our system were based on the goal of developing a low-cost, non-obtrusive, robust, customizable and portable system.

Dais location

We’ve decided that we will have a person identify the location of the dais before the recording starts as this was the most robust and feasible means.

Lecturer detection and tracking

For detection we will use an upper-body Haar classifier to initially detect the lecturer and based on this information, we’ll track the lecturer using color detection and motion detection with heuristics on possible lecturer movement. We did not favor special markers because of their obtrusiveness. Running an expensive function such as the haar-detector iteratively is not feasible.

Board Contents

We decided to go for the lower cost, robust solution and use our lecturer tracking camera to periodically take shots of the board.

Slides

We’ve decided to use a PowerPoint plug-in to capture slides and slideshow transitions.

General Audience Location

Our system will require a person to initially mark out areas where the audience will be sitting, we can use this area to capture wide-angle views of the audience and we’ve decided to use a face detector before zooming-in on the audience to ensure we don’t get empty seats.

Active Audience Location

To capture audience participation we’ll use a microphone array capable of sound-source localization.
CHAPTER 2

DESIGN
Figure 1. Data Flow Diagram Level 0

Automated Video Recording of Lecturer - AVRiL
Automated Video Recording of Lecturer - AVRiL

Figure 2. Data Flow Diagram Level 1
CHAPTER 3

IMPLEMENTATION
System Components

This section introduces the components of AVRiL while giving more attention to the engineering aspect of the project. It will describe in detail the roles and interconnections of both the hardware components and the engineering related software components in detail.

At the lowest level we have cameras that feed video to (hardware) digitizer cards. There is also a microphone array which passes sound streams to sound cards for audio processing. The videos are passed above to the Lecturer Tracking Module for detecting the position of the lecturer; and the sound streams are passed above to Audience Tracking Module. All processed video streams are passed to the Direction Module which selects appropriate streams and encodes data into a single video stream. The encoded video is sent to the Presentation Compiler Module, which compiles the video with all other presentation data like slides into a format which can be easily played back on the Presentation Module.

Hardware Level

Cameras

The whole process of video capture involves a number of engineering components. The lower hierarchy of the project is pre-dominantly occupied by engineering hardware components. If we observe the data flow of the system envisaged, one can draw a conclusion not only for the Video Capturing modules, but also for the rest of the system, that at the lower level there are engineering systems involving hardware components which feed data to upper layers having software components. Furthermore in the case of Video Capture, there is a feedback system within some hardware and software components, as it will be explained below.

For further explanation look at Camera Module Section on page 17.

Lecturer Tracking Camera

This camera will be installed to also act as the Wide-Angle Camera. If all modules work well, this camera should provide a video stream of the tracked lecturer on the dais as well as wide-angle shots of the lecture hall now and then. This camera is attached to a PTZ stand which aids the control of its motion by software layers above. For proper working of this camera two things are to be be insured; that there are no occlusions between this camera and the lecturer; there are no other dynamic objects (in the form of other lecturers or even in the form of, lets say, moving machine parts) on the dais (refer to Requirement Specification, Section 2.4).

Audience Tracking Camera

This camera is installed facing the audience; in a location where the entire lecture hall can be viewed e.g. directly above the projection screen. It basically works in
conjunction with the Audience Tracking module to roughly give video streams of locations where sounds from the lecture hall could be possibly originating from. For proper working of the Audience Tracking Camera it is important to constrain the audience to ask questions one at a time, which also make pedagogical sense (refer to Requirement Specification, Section 2.4). This camera is too attached to a PTZ stand which controls its FOV. Secondly, as mentioned above, this camera too generates analog output which has to be converted by a digitizer.

**Microphone Array**

This component is built using a set of microphones facing the audience. The purpose of this module is to detect to a vague extent the location from which sound is originating. It will be used in situations where the audience asks questions from the lecturer and the system tries to estimate (vaguely) where the person, questioning, is sitting. We envisage using multiple standard microphones placed at strategic locations which can be used to for audio localization. A simple everyday example can be the stereo microphones installed on some video cameras; you could use sound streams from the microphones and tell by the intensity of the sound from the respective microphones to estimate if either the sound is originating from the right or left.

This component is still under development. For further discussion look at Audience sound source localization Section on page 46

**Raw Data Level**

**Video-Receiver**

As the digitizer cards receive raw analog video signals from the cameras, they convert the signal to digital video streams for processing. These digital video streams in the form of frames are processed by the Video Receiver. At the initial look, one might feel that there is no need for the Video Receiver Module and frames could be directly passed to the Lecturer Tracking Module or the Audience Tracking Module. But the Video Receiver Module is essential for two things: for bringing all frames of video to set common parameters e.g. to bring all frames to a standard brightness level (since different FOV could have different brightness levels); for reducing the frame rate for the Lecturer Tracking Module because it might not be required by the latter to process all frames so the Video Receiver Module will drop some amount of frames and pass the rest to the concerned Tracking Module.

**Camera Controller**

The Camera Controller Module either collaborates with an Audience Tracking Module or a Lecturer Tracking Module to control the PTZ parameters of an Audience Tracking Camera or a Lecturer Tracking Camera (respectively). The Camera Controller takes input from the respective Tracking Module in the form of parameters telling the
amounts by which a camera has to be rotated or zoomed. The Module in turn communicates with the serial port to send instructions to the PTZ Stand of the Camera. For further discussion look at Camera Control Class on page 21.

**Audio Processing**

The Audio Processing Module is analogous to the Microphone Array as the Video Receiver Module is to the Cameras. This Module gets its data from the sound cards connected to the microphone array. It does some “preparatory work” on the audio streams and then passes it to the Lecturer Tracking Module. The nature of the preparatory work is of the same nature as that of the Video Receiver, e.g. that if the Audience Tracking Module wants sound streams in a lower bit format than that produced by the Sound Card itself, it will be the onus of this Module to produce audio streams in the required bit rate. It might also be the responsibility of this Module to produce certain processed sounds from certain sections of the stream which might be more useful or more easily processed by the algorithms implemented in the Tracking Module.

**Software Tracking Level**

**Slide Transition Detection (also part of Raw Data Level)**

Detects, stores and communicates (to the Direction Mixing Module) the times of slide transitions and the snapshots of the slides themselves. (Refer to the Design Specification Section 2.1.4)

**Physical Design Rules**

This Module is helpful to all the Tracking Modules and the Camera Controller because it contains all the information of the locations and FOV of all the cameras with respect to their physical location inside the lecture hall. All the decisions taken for deciding the parameters by which to move cameras are taken after referring to this Module. (Refer to the Design Specification Section 2.1.1.d)

**Lecturer Tracking**

The job of this Module is to (1) track the lecturer and her/his movements in the video stream (2) instruct the Camera Controller Module to move the camera accordingly (while referring to the Rules Manager Module) (3) and pass the video stream to the Mixer of the Direction Mixing Module. Please refer to Lecturer and Audience Tracking Section on page 24 for further detail.
Director Level

**Direction Mixing**

The Direction Mixing Module has two tasks to handle (1) to use good direction heuristics [1] [3] [5] to instruct Tracking Modules on Camera motions (2) to use the same heuristics to decide on the best shot to choose from the input of the three video streams. (refer to the Design Specification Section 2.1.5).

For a detailed discussion refer to the **Direction module** Section on page 38.

**Rules Manager**

The task of the Rules Manager is to basically collect useful information (like changes in slides or question raised from the audience) and to instruct other modules on the given direction rules to create better video streams. The Module can be compared to the mind of a movie director (refer to the Design Specification Section 2.1.5.a).

**Mixer**

The task of this Module is to choose the most appropriate video stream from the input set of three video streams. This Module can be compared to the actions of a movie director (refer to the Design Specification Section 2.1.5.b).

**Encoder**

The task of the Encoder Module is to encode the video stream chosen by the Direction Mixing Module into an appropriate video format (refer to the Design Specification Section 2.1.6).

**Data Compilation Level and Presentation Level**

(refer to the Design Specification Section 2.1.7)

**Presentation Compiler**

The task of this Module is to compile the given video and the presentation slides in to a set format which can be easily played back by the Presentation Module.

**Presentation**

The task of this Module is to read the output generated by the Presentation Compiler Module and to be able to replay it at any given time.
Camera Module

AVRiL has been intelligently divided into real life entities, in order to cover the workings of a video making crew. With essential divisions of cameraman and the director, the system has been further divided into mandatory software components so as to fix the whole mechanism in a method that a computer can comprehend. Illustrating on the system components, the
system is divided into 5 main classes, namely serialComm, ptzCameraControl, cameraMan, tracker, director.

**Setting up the Camera in an auditorium**

If the system is to be set in a new location, a profile of the auditorium has to be created using the AVRiL wizard. The wizard gives the user options of the basic shape of the auditorium, and the dimensions of the room, and then asks for the number of the cameras placed along with their respective placement. Further the wizard recommends the user to specify the area where the instructor will initially be found and where the audience will be sitting.

This would create a profile of the auditorium for later use; any recordings done in this auditorium afterward won’t require any specification and settings, only selecting the profile for the room would be required to start recording.

The **camera controller module** caters to the persona of a camera man and acts according to its bona fide character. Its tasks include receiving and interpreting instructions from the director and acting accordingly by conveying messages to the ptz camera. The camera controller module is a superset of cameraman class, camera control class and the serial communication class. This puts the camera controller module lower down in the hierarchy in comparison to the tracker and the director class.

![Figure 4. Different shapes of Auditoriums](image)
Figure 5. Elmo camera used in AVRiL

Workings of the respective subclasses are described below;

**Camera Man Class**

In bounds of the camera controller module, this is a higher level class with functionality restricted to translation of instructions received from the director in a format to fit any camera.

A camera man object, similar to cameraman in real life, controls one camera and takes its sole responsibility. Virtual director communicates directly with the camera man to take action, in order to achieve the results desired by it. Along side it is worth mentioning that the director has no software restrictions on having a maximum number of camera men working for it. Although there is a hardware restriction on the number of cameras that can be daisy chained therefore depending on the type of the camera coupled with the system, the restriction varies from 7 cameras to 223 cameras that can be controlled simultaneously by the director.

Further the director passes down instructions to the camera man in the form of a rectangular box, or 2 coordinates of the top left corner and the bottom right corner of the rectangle. This rectangle box is the final frame that the director wants to appear from the camera after its motion relative to the current frame.
The duty of the camera man is to process these coordinates and translate the values of pan, tilt and zoom to the camera with calibrations for the specific camera this camera man controls. The camera man when instantiates, checks for the camera it is handling, and calibrates its self with the camera accordingly. Similar to a real life camera man, who on different cameras has to vary his or her style adjusting to its movements and functionality, AVRiL’s camera man also adjusts itself with the precision of the movements it would have to make when it needs to move or focus. Essentially the camera man maps values of pan, tilt, zoom and speed for the camera with the decisions it makes from the rectangle coordinates its receives from the director.

- For the Sony camera the camera man sends different commands where as for the Elmo camera it has to send different commands

**Take action**

Take action is a function of the camera man class which is responsible for calculating the move the director wants and sending its calibrated coordinates to the camera controller class.
- current frame
- final frame
- Both with (x, y) coordinates

While taking action the camera identifies the center of the final rectangle and calculates the movements required to move the camera to the final location by finding the

---

**Figure 6. Frame zooming**

![Figure 6. Frame zooming](image)
difference between the center of the current frame and that of the final frame. This brings the camera to the center of the required frame. Thereafter the zoom increment or decrement required to fit the rectangle to the final screen is calculated. If the final rectangle is of proportion of video resolution then a ratio of required frame and that of the video resolution or current frame is calculated.

- current frame
- final frame
- Ratio of resolution

Next the zoom level is calculated just by the ratio of the final and current frame. If the ratio is equal to 1 no zoom is required, if it is less than one then we need to zoom in, and if greater than one then zoom out. But in case the ratio of the rectangle is not in proportion to the video resolution then the zoom level is calculated on the longer side of the rectangle. Upon getting the pan, tilt and zoom values they are passed down to the camera control class after these values are mapped on with the camera calibration.

The camera man also provides the director with a degree of control on the camera it is operating so that when the director loses confidence on the rectangle it won’t have to call the take action function. And consequently instructs the camera to zoom out, move to home destination of the camera, or move to a defined preset of the camera which the operator has defined. This explains how the camera man is designed to receive instructions from classes higher than itself.

**Camera Control Class**

There were different ways in which the camera could be controlled. Different commands that could be send to the camera.
Camera could be controlled by sending it absolute coordinates or relative movement commands.
Variable speed for both pan and tilt
When the video feed of the camera is being recorded the camera is made to move slowly
Otherwise when its video feed is not captured for example in the case of the audience camera, the camera is made to move faster to get to the destination quickly so that it’s switched to faster.

A separate class was made in order to have the liberty to use different makes and types of cameras without having to change the programming to a great extent. This is a plug and play class, such that for each make and model of camera compatible with AV RiL, respective class would be used so that no other programming modifications are needed. According to the camera used with the system, camera control would send instructions to the camera for that particular make and model.

Achieving this proposal was a tricky task, from defining demarcations and separating responsibilities to laying down the versatility and functionality we wanted from the
system. And with no compromise this was achieved by the introduction of a camera control class separate from the camera man class.

From basic to higher level functions the camera control class is designed to be versatile yet robust.

```cpp
ptzCameraControl();
/*//////////////////////////////////////////////////////////////////*/
~ptzCameraControl();
/*//////////////////////////////////////////////////////////////////*/
//SETs
int setTiltSpeed(int speed);
/*//////////////////////////////////////////////////////////////////*/
int setPanSpeed(int speed);
/*//////////////////////////////////////////////////////////////////*/
int setPan(int pan);
/*//////////////////////////////////////////////////////////////////*/
int setTilt(int tilt);
/*//////////////////////////////////////////////////////////////////*/
int setZoom(int tilt);
/*//////////////////////////////////////////////////////////////////*/
int setPixPerPanTilt(double panPix movePerDeg, double tiltPixtmovePerDeg);
/*//////////////////////////////////////////////////////////////////*/
//ZOOM FUNCS
int zoomDirect(int zoom);
int zoomRel(int zoom);
int zoomIncrease(unsigned int units = 150);
int zoomDecrease(unsigned int units = 150);
int zoomPosInq();
/*//////////////////////////////////////////////////////////////////*/
//MOVE FUNCS
int moveRel(int unitsPan, int unitsTilt);
int moveRelDeg(double degPan, double degTilt);
int moveRelPixel(int panPix, int tiltPix);
/*//////////////////////////////////////////////////////////////////*/
//RESETS
int resetPanTilt(void); //rests Pan and tilt
int homePanTilt(void);
```

**Serial Communication Class**

Last in the camera control hierarchy
Two way communication
Send and receive data packets through the serial port rs 232 port.
When a number of cameras are daisy chained, from 1 serial port we can control all these cameras.
The camera sends ACK and NACKs, both received through the serial port to check the status of the camera from its response.

```
serialComm();
/*//////////////////////////////////////////////////*/
~serialComm();
/*//////////////////////////////////////////////////*/
int send(unsigned char *msgBuff, int msgSize, DWORD &dwBytesWrote);
/*//////////////////////////////////////////////////*/
int recieve(unsigned char *msgBuff, int msgSize, DWORD &dwBytesRead);
/*//////////////////////////////////////////////////*/
```
Lecturer and Audience Tracking

This section explains how the system understands the important part of the classroom / lecture environment. It most importantly involves understanding the position of the lecturer in the view of the camera. Secondly it also requires the detection of the audience for two reasons as given below. First this section will explain fully the motivations behind tracking a lecturer or people in the audience. Then it will move to what problems the trackers can face in a lecture environment. Then the two methods used for tracking, Motion Tracking / Motion Histogram are detailed in the document. For both the methods, first the techniques involved in making the tracker i.e. the actual nuts and bolts behind the working of the specific tracker are explained. The actual classes as used in the system’s code are also explained with each tracking method.

Motivation behind tracking

In the system developed, tracking forms a major part of the way AVRiL tries to understand the environment it is in. Without it, effective direction would be almost impossible. It is important to note that without a working tracker, direction is still possible in a way by deciding intelligently when to change the shot between the cameras. However camera panning and zooming are only made possible by a tracker. Even though it might be safe and logical without a tracker to take wide-angle shots of the lecture hall, a bad directed video might result if the tele-view of the camera cannot cover the whole dais (the area used by the lecturer) or in cases if two or more cameras are used to cover a part of the dais.

A lecturer tracker gives many important cues to the director. A good tracker can notify the system that the lecturer is moving out of the camera’s view, hence enabling the system to intelligently pan the camera. A tracker can also allow the director to check the amount of movement a tracker has gone through, allowing it to decide when to zoom or not. As for audience tracking, without a tracker the system can only obtain a wide-angle shot. A tracker can identify individuals in the audience for whom zoomed in shots can be taken, which creates a much more interesting video rather than just change of wide-angle shots. Other cues that could be possibly generated with a tracker are the actions of the lecturer which could be used in deciding gestures like where the lecturer is pointing. Furthermore a tracker can also point out that has the lecture started or ended; important signals for an automated recording.

Although a good tracker is quite difficult to build, it makes the output of the system seem more intelligent. Without a tracker it will be quite simple to realize that the video was directed without any human intervention. As an example if there is a lecture happening in the absence of audience the video conceptually should not contain any audience shots. Without a tracker, the system will obliviously take shots of the audience, whereas in the presence of it, the system can learn that there is no requirement of an audience shot.
Problems in Tracking

There are several problems in doing effective tracking. It is a deceptively hard problem since it’s a trivial for the eye and human brain to track any object at any waking moment. For a machine it is harder especially when it is difficult to calculate who is tutoring or speaking. Apart from that any good tracker can be confused in many situations. This is because a tracker is always looking for some qualities that are distinct in humans and not discerning who is actually instructing. As an example, in a lecture hall where the lecturer is a person whose video stream is being webcast and shown using a projector might lead the tracker to believe that there is no lecturer in some cases. Below we will explain a few problems that some trackers might encounter, but the problems are in no way limited to the ones stated.

One of the common pitfalls of a motion tracker (as explained below) is that it takes cues from motion generated by anything. Hence any moving object can potentially confuse a tracker to falsely report the location of the lecturer. This is a common problem for a camera having wide-angle view of the lecture hall. If apart from the seeing the lecturer, the audience also falls in the field of view of the camera, motion generated by the audience can be greatly confusing. If you look at the two sets of frames below, the idea will be clear (Figure 7). These two frames are set apart by 15 frames, and the point to note is the amount of motion naturally generated by people in the audience. Also note the little amount of motion by the lecturer in that same time period.

![Figure 7. Two frames showing the large motion in audience](image)

Other places where motion can occur are objects used by the instructor himself to help in the teaching process. For example many instructors use slideshows through their lectures. Slides can always contain animation or videos according to the contents of the course. If a tracker is not detecting if the motion originates from some human form or not, it can also get confused by motion on slides. Analyze the two frames given below (Figure 8). If one notices the lower end of the slide, new lines appearing in the animation will become apparent. Also note the small amount of movement by the lecturer standing adjacent to the slides. A tracker blindly noting motion will of course be mislead in such situations.
Other problems that can potentially hinder the tracker are complete changes in the whole environment itself. Such changes can be brought about by camera movement. When a camera moves i.e. pans, tilts or zooms, for a machine seeing just pixel values, the complete surrounding changes. In fact a large amount of such global motion can be hazardous for a motion tracker. So if during the motion of the camera, a motion tracker is operating, it will get confused from the large amount of motion generated throughout the frame. Look at the two frames given below; one before and the other after the camera movement (Figure 9). The scene of the lecture hall between the two frames has drastically changed. As explained later, it is also a difficult for a motion tracker to estimate the new position of the lecturer after the movement without re-initializing.

Other changes in the lecture hall which change the complete scene are lighting changes. Many lecturers vary the intensity of lights to make things on slides more legible. This action should not affect good trackers, but just like global motion generated from camera movement, lighting change also results large change in pixel values throughout the frame. Note the effect of change in lighting between the two frames given below (Figure 10). Both the event of change in lighting and low light condition can result in bad output from
a tracker. The event of change in lighting, as explained above, can fool a tracker because of the large change in pixel values, whereas low light condition can make every object of importance very vaguely visible and hence difficult to track.

As explained in the next section, if we are just doing background subtraction, any significant change in the environment can hinder the process of extracting useful information. Take the example of a lecturer who writes excessively on the board and also rubs it to write more. This will surely generate some change in the board itself i.e. the background. Any change in background can be a hindrance in tracking in a basic background subtraction tracker. This condition also holds true for boards that can be moved like those available in lecture halls at MIT (Figure 11). This will have the same effect and create the same problems for a tracker as writing new content on the board, change of slides and new objects coming into the environment.

A tracker can be greatly helped by distinct clothing of a lecturer. If the lecturer’s outfit makes him stand-out from the background, it helps color trackers to a great degree and motion trackers to a smaller extent. But if the lecturer wears a color similar to its
background, like a dark green shirt while teaching on a black-board, or a white shirt in front of a white slide drop, it can be confusing for a tracker. Look at the two frames from different lectures given below (Figure 12). The distinct blue color in the left frame would greatly help some trackers while the off-white shirt of the lecturer in the right frame might be confusing especially due to the white back-drop and the off-white walls.

Other problems for a tracker can be due to other circumstances. One of them is the presence of multiple lecturers. AVRiL supposes that only one instructor is present in the lecture hall and currently does not solve the multiple lecturer problem. This is a great pitfall for some lectures. We believe that an environment with more than single lecturer cannot be solved by a simple tracker. It might be solved by obtaining sound cues, face detection, or using computer vision to detect the speaking lecturer.

Motion Tracking
AVRiL relies heavily on a motion tracker for detecting a lecturer. It uses a motion tracker built on the technique of motion histograms, which will be explained in detail below. Motion tracking is basically a method which involves frame differencing to get motion over the last frame change, and then a detection step of pin-pointing the location of the biggest cluster of motion pixels. Motion tracking is an easily implemented tracker and a good choice where motion can be localized to the object needed to be tracked.

Technologies and Techniques

Change Detection / Frame Differencing
Motion tracking can be essentially done in two ways. You can either do a background subtraction or frame differencing. In background subtraction the wide-angle camera would first take an initial shot (or a set of shots) before the lecture starts in-order to build the background of the lecture hall. Then over every subsequent frame in the lecture, that frame would be minus-ed from initial
background frame. This is not such an effective method as any changes in the lecture environment will result in the subtracted image i.e. any new object coming in like the lecturer opening her laptop. The concept of background subtraction is given in the image below (Figure 13). Although the background model is built with a Gaussian of a set of background images, the final subtracted output has a lot of noise in it.

\[
\nabla_{\text{pixels}} | \text{frame}_i - \text{background}_{\text{model}} | \geq \text{threshold}
\]

(1)

Figure 13. Background Subtraction run on a background image and a frame taken during a lecture

Rather than doing background subtraction, which is bad when the background has changed by a great degree, we do frame differencing. Rather than subtracting each
frame from the background model, you subtract each frame from every previous frame (Figure 14). Hence for every frame the following formula is used to obtain the foreground image for that:

$$\nabla_{\text{pixels}} \mid \text{frame}_i - \text{frame}_{i-1} \mid \geq \text{threshold} \quad (2)$$

There are also better ways to do frame differencing. You can make a better background model by using a weighted average of the previous few frames as follows.

$$B_{i+1} = \alpha * F_i + (1 - \alpha) * B_i \quad (3)$$

The value for $\alpha$ is set typically to a value close to 0.05. The foreground is computed in the same way as before, by just subtracting the background from every frame. The effect of this equation would be to create a background which is built by some percentage from the previous frame and increasingly smaller percentage from all the frames before it.

Figure 14. Frame differencing operation illustrated
**Gaussian Masking**

Gaussian masking is used to denote at least two things when we talk about tracking from images. The first is used in the context of building a Gaussian background model. In this method, a group of background images is taken and the mean and variance for each pixel is calculated. Once that is done every new frame’s pixels is compared with the mean, variance and a specific threshold to calculate if the pixel forms the foreground in the frame or not.

AVRiL specifically uses a Gaussian mask for subtracting two frames. When using a Gaussian mask, foreground calculation for a specific pixel is not only dependent on the pixel values over the frames at that location, but also weighted over a pixel region around. This pixel region is modeled around a 2D Gaussian curve as given in the figure below (Figure 15).

![Figure 15. Shape of a 2D Gaussian for taking weighted average of pixels](image)

**Lecturer detection**

It is important to note that a lecturer cannot be only detected by just doing some form of background subtraction or frame differing. A detection step is necessary to complete the process. Think of it this way that once frame differing has been done, the system has figured out the foreground. The story doesn’t end there as foreground might have a lot of other things than the lecturer himself, since other things in the lecture hall can also be moving. So from the foreground, the system needs to detect a boundary where the lecturer possibly stands. Maybe by some complex means, you could carve out an exact boundary of the lecturer but since this process will be done tens of time in a second, it needs to be as computationally cheap as possible. Hence, just computing a rectangle fit suits our purpose the best. From intuition, one can clearly say that marking a human silhouette would be much more difficult than marking a rectangular bounding region. The whole lecturer detection process is explained in the instructive figure below (Figure 16).
Figure 16. Steps of Lecture Detections

Step 1: Frame Differencing

Step 2: Gaussian Masking

Step 3: Motion Histogram

Step 4: Thinnest Mountain

Step 5: Mark Lecturer in accordance with History
So how is this rectangular bounding region marked? After frame differencing (to mark the foreground) and Gaussian masking (to enhance the foreground), a Motion Histogram is built of the frame. The motion histogram method is extremely simple. It just divides the frame into multiple vertical columns where each column is around 5 pixels wide. We build a motion histogram by noting all the non-zero values in each column i.e. the histogram has as many values as there are columns and each value represents the number of non-zero pixels in that column. If you add up all these values in the histogram, you will get the total number of non-zero values in that frame which we will call ‘Total Motion’. One should note that we are not talking about the total number of non-zero values in the original frame rather we are talking about the non-zero values remaining after the process of frame differencing.

\[
H_k = \sum_{i=\text{column width}*k}^{(\text{column width}*(k+1))-1} \sum_{j=0}^{\text{frame height}} P_{x(i,j)}
\]

\[
\text{Total Motion} = \sum_{k=0}^{\text{frame width} / \text{column width}} H_k
\]

The value of Total Motion is an important one. Before creating a histogram of the given frame, the value of Total Motion is checked to see if that it is within certain limits. If it exceeds a certain threshold the frame is not processed and it is supposed that the lecturer hasn’t moved. This is done since a large number of motion pixels can be generated by change in lighting, animation on slides, camera movement and many other things. Translation of lecturer during camera movement is dealt through other methods.

After building a motion histogram, all that needs to be calculated is the biggest cluster of motion values. To calculate this we use a method called the “Thinnest Mountain” method. In this method we try to figure where is the bunch of columns which contains 60% (or some other higher percentage) of all motion pixels. Figure 16 illustrates the thinnest mountain method. In the figure the dark grey bars show the part where the lecturer stands. The two vertical markers show the end result of the Thinnest Mountain method. The diagram points out that it is always not possible to get completely correct results, as according to the motion over the last frame, some portions of the lecturer might be left out of the bounding region computed. As for computing the horizontal boundaries, we can do the same method but less frequently, as we don’t expect the lecturer to move much towards or away from the camera as much as he would move across the frame.

**Filtering motion noise (camera / slides / audience motion)**

As stated before apart from the motion generated by the lecturer there can be many other things in a lecture hall environment that can create motion. One of them is the audience. One very important aspect that can be exploited to deal with audience motion is the fact that all the audience appears to be lower to the lecturer when
viewed from the lecturer tracking camera. To cater with this audience noise, you can ignore the motion pixels in the bottom part of the frame while building motion histograms. Other noise can originate from slides in multiple ways. For example during slide change, over one or two frames this action will generate large amount of motion pixels. This will be simply dealt by the method stated earlier in which frames with a ‘Total Motion’ larger than a certain threshold are ignored and the lecturer is supposed to have not moved. This is a good enough supposition because even if the lecturer has moved over a few frames, the motion will not be large enough to completely de-rail the tracking process when it finally continues.

There can also be motion noise generated by activity on the board. Chalk (or for that matter a marker) writing by a lecturer is not a big problem since over a time period of two frames, the board content would not have changed that much. This means that change in board content would not generate many motion pixels. But as for the movement of the board itself, it will be treated as too many motion pixels generated which will cause the frame to be ignored.

Software Engineering

TrackerAlgo class
The TrackerAlgo is a base class from which all the tracker classes inherit. It defines the basic class structure that should be of tracker class, like it should provide functionality for another class to basically call a tracking function on each frame trackerLoopIteration(). This allows another function to fetch frames from any device and hand it over to the tracker. This allows the TrackerAlgo object to decide the change in position of the lecturer by extracting information from the frame given and also information gathered over the previous frames. There is also a function findLecturer() which is used to identify the lecturer in the beginning. To initialize the tracking we also have a function initAll() whose functionality is built by the inheriting class.

MotionHistogram class
The MotionHistogram class encapsulates all the functionality of the base class TrackerAlgo. Most of the functions in TrackerAlgo are kept pure virtual i.e. we have to define the function inside this new inheriting class. For example we have to define initAll() internally in this class because unlike the standard functionality defined in TrackerAlgo only, you also need to define data structures which would eventually hold the histograms and the related information.

MotionHistogram class also provides functions to display the motion histograms as images for the purpose of analysis. This class returns the position of lecturers also according to the previous frames given to it. If across two frames the position of motion has changed greatly, the new motion values are ignored, and it is supposed that the lecturer hasn’t moved over the last frame.
Tracking with the aid of Color

Technologies and Techniques

Color tracking
The model of color tracking used in OpenCV is called the MeanShift initially developed by Comaniciu. It basically revolves around the idea that color can be used to track objects. The feature it lacks is the power to initiate detection. And so it will seem intuitively, since unless the tracker exactly knows what color to track, how will it even start its tracking. So the use of a color tracker is meaningful only when you couple it with a motion tracking algorithm, where the latter first initiates the tracking, and the former probably carries on the tracking.

The MeanShift algorithm outputs basically a mean point in the data and over consecutive it tries to calculate the shift vector for this mean point. The idea here is that the mean’s shift vector would always point to the increase in density. The target model for tracking is to calculate the feature probability distribution by using weighted histograms. For a detailed explanation look at APPENDIX B: Object Tracking through MeanShift.

Software Engineering

MeanShiftTracker
Just like the MotionHistogram class, the MeanShift class also has some added functionality on top of the base class. First of all, just like all trackers, it also does the computation every time you provide it a new frame. The difference here in this class’ initialization is that you have to provide the region of the lecturer in a frame. This is done by either use a Haar Classifier (as explained in the next section) or by using motion analysis using the TrackerAlgo class. The MeanShiftTracker’s trackerLoopIteration() function return just the shift in the lecturer position i.e. the new lecturer position. It also provides two other functions that can set parameters called the vmin and vmax which can by used to fine-tune the tracking for a given lightening condition.

Tracking through Trained Classifiers

Technologies and Techniques

Haar Classifiers for profile face / upper torso
OpenCV provides a very robust algorithm for any sort of object detection. This is possible through Haar like wavelets, which allows matching objects against trained data. In the set of pictures used for training, the algorithm tries to record features present in every picture. The algorithm records features by testing in what places a
specific feature occurs by matching it with patterns of black-white tiles. This allows the algorithm to be extremely quick in reducing the picture to areas which could possibly have these features and hence possibly the object. For a more detailed explanation of how Haar Classifiers work, read APPENDIX APPENDIX A: Object Detection through Haar-Classifiers in OpenCV.

We use Haar Classifiers for two purposes. We can do initial lecturer detection either by motion analysis or by using a Haar Classifier. OpenCV comes with a trained data set of Haar Classifiers for many types of faces and with body shapes. One of them is the human upper torso, which is pretty effective in detecting the lecturer since the only upper torso usually visible from the wide angle tracking camera is that of the lecturer. Usually only heads of the audience are visible as their bodies are occluded by seats. The purpose we use Haar Classifiers is for detecting audience from the audience camera. AVRiL takes zoomed in shots of the audience pretty often. For this, first the audience camera runs a Haar Classifier for faces on a wide-angle view. This will hopefully mark all the audience the camera can see, from which the system would choose a person randomly for a close-up shot.

Software Engineering

AVRiL uses two modified versions of this class. One is for initial lecturer detection through upper torso Haar Classifiers, and the other is for audience detection.

HaarClassifierLecturerDetector class

This class is used when the recording just starts. The programmer or the set-up manager of the system can choose either to TrackerAlgo class’ findLecturer() method for initial detection of the lecturer or use this class. Our testing shows that there is a high chance that this Haar Classifier method for upper torso detection might fail if just used once. To effectively use it, it has to go through several passes to be confident of the position of the lecturer. The other drawback of using this method is that it is really slow. It takes around 1-2 seconds to complete a Haar Classifier run on a 640*480 frame, which are light years in video processing. Nonetheless when used, it provides a findLecturer() method to detect a lecturer. What it does is run the upper torso Haar Classifier on the frame it already has. It returns either a rectangle of the lecturer position or just a false value indicating failure. It also passes a false value when it detects more than one upper torso. This method is at-least called thrice to be sure of the position of the lecturer.

HaarClassifierAudienceTracker class

This class is used by the system to take shots at random the audience. What we don’t want is that from the audience camera we take zoomed in shots of empty seats. Rather we want a more engaging video at the end which possibly show the expressions of the audience.

This class gives a function findAudience() which is called whenever a random person needs to be selected from the wide-angle audience shot. This function returns
immediately and runs the Haar Classifier on a separate thread since the process
takes pretty long and during video recording the system tries to avoid any large
delays, which will make the recorded video jittery. One the function has returned
the system can check the status of the results by using a function called
returnResults(). This function can return four things; it can return a
NoProcessingRequested which denotes that no task has been assigned as yet, it
can return a positive result and return a randomly rectangle from the audience, it can
return a NoResults which denotes that no audience was found giving important
clues to the system that it shouldn’t take even a wide-angle shot of this location, and
finally it can return Processing which just tells the caller function that the Haar
Classifier is still processing the frame to find any audience.
Direction module

The Direction module handles one of the most innovative and interesting parts of the project. This module is responsible for making sure that the video produced by the AVRiL system is interesting, engaging and professional, while at the same time capturing the complete classroom environment as best as possible to give the viewer the best possible classroom look-and-feel, which is the main aim of our project. To achieve this goal, the Direction module uses an approach of implementing cinematography rules into code. Each rule’s requirements cause the Direction module to grow in terms of variables, functions and even sub-classes.

Note: In the rest of this section, the “Direction module” is also referred to as the “Director”.

Responsibilities

**Camera Input Selection**

The Direction module decides which camera feed is going to be used for the final video, at different points during the lecture. Cameras need to be chosen intelligently, and a balance needs to be maintained in the gap between camera switch times, in order to make sure that the video is neither too boring, nor becomes unwatchable.

**Camera Movement Commands**

All commands to the PTZ cameras (CameraMan class at the software level) are also sent exclusively by the Direction module. There are also a lot of direction rules applicable here, which ensure that camera movement is minimized during the time the camera has been selected for final video output.

**Implementing Cinematography Rules**

The Director module, as mentioned in the introduction, is also responsible for the application of professional cinematography and direction rules and heuristics to the video being produced. This gives the Direction module the task of converting rules that are understood by humans relatively easily, into code that a computer can understand. The applied rules are given in the following section.

**Cinematography Rules**

Here is a list of the cinematography rules that AVRiL implements:
**Rule 1: Switch between cameras**

One very basic rule of video recording, whether it is of a lecture, a stage show, or anything similar, is to frequently change the camera view. This is commonly achieved by switching to a different camera if multiple cameras are being used. Meanwhile, the previous camera can also change its pan-tilt-zoom levels to achieve a new angle on the lecturer or the audience. The time between transitions varies depending on a lot of variables, including the current shot context, audience participation, and lecturer activity level.

**Rule 2: Minimum time-period for a camera-shot**

Another precaution to take when implementing Rule 1 is that cameras should also not be changed very quickly. If this is done, the video will become confusing and irritating for the viewer.

**Rule 3: No camera movement during a live camera-shot**

This rule of videography says that when a camera is selected for video output, ideally it should not be moving. This necessitates setting the camera pan, tilt and zoom values to such a level that keep the video interesting while making sure that the camera will not have to be moved or switched very soon.

**Rule 4: Lecturer moves out of the camera view**

When the lecturer moves out of the frame, the Direction module has two choices; one is to switch to another camera that can capture the lecturer at his new position, and another is to follow the lecturer with the current camera by changing its pan. This decision is made based on the speed with which the lecturer exits the frame. How this is done is explained in the Technology and Techniques section.

**Rule 5: Camera-shot of a participating student**

Rule 5 is one of the main rules that gives the system the classroom look-and-feel. It says that whenever an audience member is participating in the class discussion, a suitable camera should be selected and panned and zoomed to show that particular student who is participating. Other students may also be a part of the frame recorded.

**Rule 6: Intermittent camera-shots of audience**

This rule aims to make the video more interesting and engaging by including shots other than the lecturer. It says that even if there is no audience participation for some time, the video should switch to a random audience member or a group of audience members in order to make sure that the video does not get monotonous. The time given to this shot is usually very low, i.e. around 5 seconds.
Rule 7: Lecturer positioning in the frame

In lecturer shots, an effort should be made to keep the lecturer at or around the center of the frame. This helps to keep the video single-minded and retains the focus of the viewer on the lecturer.

Rule 8: Zoomed-in view of the lecturer

This rule also tries to make the video more interested and engaging. It says that if possible, shots of the lecturer should be as zoomed in as possible. There is obviously a maximum acceptable zoom level, which is zooming in enough to show the lecturer’s head and torso. Keeping in mind that camera movement also has to be minimal (Rule 3), this rule needs to be applied carefully, considering variables like lecturer activity level. This is explained in Technology and Techniques section.

Rule 9: Lecturer positioning in zoomed-in camera-shots

When taking a close-up shot of the lecturer, it is important to leave around 1/8th of the screen above his/her head. Too little or too much space in the frame above the head means that the video has been poorly directed.

Technology and Techniques

This section describes some of the main techniques used by the Direction module in order to achieve the above-mentioned goals and responsibilities, including cinematography rules.

Timers

The Direction module uses a large number of timers to keep track of time passed since an event. For example, the “Preset Timer” keeps a record of how long it has been since the camera view was changed. An upper limit on the Preset Timer ensures that Cinematography Rule #1 (see Cinematography Rules section) is implemented.

Timers are also used to keep track of things like how long it has been since an audience shot was shown, the total length of the video, how long a camera has stayed still, etc. All this data is used to make decisions about camera angles, cinematography rules application, transitions and their timings, and camera zoom levels.

Safety Box

One very simple, yet powerful tool used in AVRiL is the use of a “safety box”. This is a rectangle drawn around the lecturer, covering 70-80% of the screen. This rectangle is meant to minimize camera movement, and to leave a cushion for the tracker in case the speaker is moving out of the frame. This is useful because without a safety box, the camera would only move when the speaker moves out of the whole frame, in which case the Director would not have the time to figure out the speed of the lecturer, and
would have to make sudden, fast movements to the camera which go against the ethics of making a good video. Also, this would cause the Tracker module to lose track and would have to search the frame again, after the camera has moved, for the lecturer. This reduces the overall accuracy of the tracker.

**Lecturer Speed Detection**

As mentioned in Rule 4 (Section *change*), in case the lecturer moves out of the camera’s view, the Director has two options: whether to follow the lecturer or to switch to another camera. In reality, the system does not wait for the lecturer to move out of the frame, but the safety box. The system is calculating the speed of the lecturer at all times. It is calculated by taking a snapshot of the center of the lecturer and comparing it with subsequent snapshots over the next 3 seconds. If the speed of the lecturer when exiting the safety box is high, the system makes the decision of changing the camera, based on the prediction that it will be difficult to track a fast-moving lecturer. In case the lecturer is slowly moving out of the safety box, the Director decides to follow him/her by panning the current camera.

**Activity Box**

As mentioned in Rule 8 of Section *change*, close-up shots of the lecturer are preferred over wide-angle shots. However, it is necessary to know whether a zoomed-in shot can be sustained for a certain period of time. This means that the Director will not zoom in too much on a lecturer who moves around a lot. To make this judgment of zoom level, we create an “activity box” around the lecturer when in wide-angle mode. This box records the maximum movement area of the lecturer based on a 30-second history. If at the end of the 30 seconds, the activity box is small (i.e. the lecturer has not moved a lot in the last 30 seconds), the Director decides to zoom in completely to cover only the head and torso of the lecturer, based on the assumption that the lecturer will maintain this behavior for the next minute. Of course, if the lecturer decides to start moving while the system is taking a zoomed-in shot, it will either zoom out or switch to another camera. Similarly, a medium sized activity box will result in a medium zoom, and a large activity box will result in no zoom, i.e. wide-angle shot.

**Confidence level**

The Director also makes use of the confidence level data provided by the Tracker module, which indicates how confident the Tracker is, that the tracking information it is providing is accurate. It is basically a measure of the Tracker’s perceived accuracy of its own algorithms. The decision of taking the close-up shots mentioned in Rule 8 of Section *change* also take into account this confidence level. If the confidence level is high, the Director will not hesitate to zoom in, provided other conditions are favorable. However, if the confidence level is low, the Director will hold back zooming in until it becomes high again.

If the confidence level becomes zero, which means that the Tracker is sure that it has lost the lecturer (e.g. in the case of lights being turned off), the Director will switch to
the wide-angle view, which almost guarantees that the lecturer will be in the recorded frame. The director will stay on this setting until confidence level becomes normal again.

**Camera Transition Cushion**

To implement Rule 3 of Section *change*, we need to make sure that the pan-tilt-zoom activity of a camera preparing to shoot should not be recorded. To achieve this, we provide the target camera a 3-second cushion to prepare for the shot, i.e. pan-tilt-zoom to the new setting. During these 3 seconds, the video is recorded from the current camera.
CHAPTER 4

FUTURE WORK
Thoughts on future work

We see AVRiL as an evolving project. It gives us an opportunity to look deeply into things that can further improve the system. It can be expanded and made better in many ways. For one, it can be made more robust to work in any sort of lecture environment directly out of the box. It will greatly improve the workability of the system if it can be made to work with all sorts of environments other than a lecture hall, like in conference rooms or for talk shows. Below we discuss some areas we are interested in for future work:

Web Services

To make this automated service appealing to all kinds of educational institutes, it is necessary to have effective content delivery. The service is almost of no use if it has to be manually published. When we thought about automatic educational content creation, this too comes out to be an important part of it. A result of this should be automated content publishing immediately after the lecture has been recorded and directed.

Bandwidth dictated web-services

Many universities provide services to students through which they can view all of the recorded lectures in their university. Stanford is one of them. Some of them even provide these services for anyone who wants to view them over the internet. MIT’s OpenCourseWare (OCW) has been the pioneer to such programs. Yet the problem still remains since it has never happened for any good university that all the recordings for every course offered has been made available to anyone on the web. The partial reason behind it is that it is extremely expensive for any university to record every course it has offered, yet alone make it available to everyone on the internet. It is not uncommon for a university like Harvard to offer more than 500 courses per semester. The other reason is of course that why should a university like Harvard or MIT provide its high quality content to everyone free of cost, when all of its student body pays thousands of dollars for it every year. As stated before we have aimed to solve the first problem: making quality content creation i.e. lecture recording cheap enough to allow any university to record all of their courses.

Once these lectures have been recorded, what services should be provided for their effective delivery? It is an important question to ask because without effective content delivery, the effort put in for effective content creation diminishes in value. It is important to provide web services which are not only easy to access for any student (or for only the people the university wants to) around the world, but also provides the most effective class-room look and feel without degrading the educational value of the lecture. Things that we have already been developed

Basic Web Interface
We have developed a base model for our web service which allows us to post directed videos on to the web manually. As explained below we plan to use this model for automated posting of not only lecture videos but also related content.

**Synchronized Slides**

We have also developed a framework that helps lecturers who use slides during their lectures. This framework basically records time-stamps of slide transitions relative to the video recording. This allows synchronized playback of slides with the video. This in turn enhances the learning experience since the viewer can now not only navigate through the video but also through the slides (a powerful idea since slides already have a lot of meta-data in form of text).

**Things that we plan to develop:**

**Web Service**

A more powerful expansive web-service. We are thinking of a powerful, yet basic web-service which can be used by any university to develop their own customized web-service. Ideas for the web-service:

- Option to automatically populate the web-interface with newly recorded lectures.
- RSS notifications about new lectures to subscribers.
- Authorization services for universities who don’t want to make lectures available to everyone.
- Access to all the options to save lectures for offline viewing.

**Wiki provision in web-service**

An educational web-service should be as much interactive as possible. The first question that pops up in ones mind when they think about learning from recorded lectures is about the provision of asking questions and gaining any sort of feedback from the lectures. The best way to do this is to incorporate a wiki-style service with each lecture available through the web-service. This will allow any user (or a group of users) to edit the lecture recordings by adding more data to it. Following is an extension to the list of things that we plan to build into the web-service:

- Ability to add different tags for anyone subscribing to the service. These tags might be either synchronized with the lecture or not.
  - Topic indexing. This will allow numerous people to share indexes which will enable viewers to jump to the required portion of the lecture.
  - Lecture transcription. By allowing anyone to write points of the lecture or even transcribe it, it will enable viewers to have better understanding of a lecture in case of loss in quality during transmission e.g. a viewer might choose to see just the lecture transcription rather than the video in case of small available bandwidth.
- Meta-tags. Other kinds of tags that give different kind of information about the lecture e.g. tags
  - Ability to add tags in form of queries. This will cover the part of asking questions and answering them. This can be expanded to only allow some people to answer questions.
  - Ability to tag links to the video. For example you might tag a link pointing to a similar lecture online from another university.

**Bandwidth conscious web-service**

In order to have an educational service with a huge impact it is important to have services which are aware of the limitations of a user. Thus for web services, it is a good idea to make them bandwidth conscious. By this we mean that the service is aware of how wide is the network pipeline between the client and server and what choice of content can be transferred over it.

Some ideas for providing services in cases of low bandwidth:
- Choice of providing just slides rather than video.
- Providing only the first frame of the shot and play it till any shot change. This will not leave the viewer completely clueless of the lecture environment.
- Choice of provide slides with synchronized audio without video.
- Reduce frame-rate of the video.
- Reduce resolution of the video.
- In case of wiki-styled web services, provide transcribed lectures if available.
- Provide just the board and slide content (look below at Lecture board information extraction and representation)

**Audience sound source localization**

Audience sound source localization is done to estimate where the sound is originating from the audience. This is a good cue in order to localize audience participation. We are aiming to use sound source localization in order to get vague locations of participating audience which will result in cues for pan-tilt-zoom values for the audience camera.

Currently we are working with AcousticMagic’s VoiceTracker array microphone which is capable if doing two things. First of all it detects the DOA (direction of arrival) of the sound, and second it amplifies this localized sound. Once implemented, this will throw away the need for an audience wireless microphone in a question answer session. Yet there are still other concerns for the effective use of it. Noise is one of them, as apart from audience speech, this microphone will pick up spurious noise like that of an air-conditioning unit inside the lecture hall or even the speech of the lecturer himself. This will make the positioning of this unit critical in the lecture hall. Other areas of concern are multiple speakers in the audience as it commonly happens in many classes prone to debates.
For properly integrating the device in the system, there might be need for another module. Since now there will be two audio feeds: one from the lecturer microphone and one from this array microphone, there might be need for audio mixing. This might be an important step before the audio can be integrated with the video.

Lecture board information extraction and representation

Figure 17. Initial Board writing legibility tests
The one thing still missing for creating a complete class-room look and feel is the board aspect of the lecture recording. Things chalked down by the lecturer on the board form an important part of the material communicated in a lecture in many universities. Also for a such a product to be successful from intermediate level schools to high schools, the system by no means can be effective without recording board content.

Many previous systems have tackled the problem by using electronic boards (refer to Current Systems on page 3). This solution is not only an expensive one, but also reduces the portability of the system. To tackle this problem it would be highly effective if it can be solved by Computer Vision techniques. This will allow the system to record the board content by employing an extra camera at maximum.

We have started testing techniques for board recording. A picture of the initial tests is given above (Figure 17). Once the background subtracted writing has been extracted, some image processing might be needed to make the writing more vivid, as it can be seen in the pictures that smaller text has almost disappeared. Other things that need to be done is to explore effective representation of this data. Rather than sending continuous pictures of the board on an already clogged bandwidth, we might employ some compressed writing representation. Spline-fitting is one way to go about it, but it will result in extremely complex representations in case of hand-writing.
REFERENCES


APPENDIX

APPENDIX A: Object Detection through Haar-Classifiers in OpenCV

The object detector described below has been initially proposed by Paul Viola [Viola01] and improved by Rainer Lienhart [Lienhart02]. First, a classifier (namely a cascade of boosted classifiers working with haar-like features) is trained with a few hundreds of sample views of a particular object (i.e., a face or a car), called positive examples, that are scaled to the same size (say, 20x20), and negative examples - arbitrary images of the same size.

After a classifier is trained, it can be applied to a region of interest (of the same size as used during the training) in an input image. The classifier outputs a "1" if the region is likely to show the object (i.e., face/car), and "0" otherwise. To search for the object in the whole image one can move the search window across the image and check every location using the classifier. The classifier is designed so that it can be easily "resized" in order to be able to find the objects of interest at different sizes, which is more efficient than resizing the image itself. So, to find an object of an unknown size in the image the scan procedure should be done several times at different scales. This detection process combines the following four key points:

- Simple rectangular features, called Haar features.
- An Integral Image for rapid feature detection.
- The AdaBoost machine-learning method.
- A cascaded classifier to combine many features efficiently.

The features that Viola and Jones used are based on Haar wavelets. Haar wavelets are single-wavelength square waves (one high interval and one low interval). In two dimensions, a square wave is a pair of adjacent rectangles one light and one dark.

The actual rectangle combinations used for visual object detection are not true Haar wavelets. Instead, they contain rectangle combinations better suited to visual recognition tasks. Because of that difference, these features are called Haar features, or Haar like features, rather than Haar wavelets. Figure 18 shows the features that OpenCV uses.
The feature used in a particular classifier is specified by its shape (1a, 2b etc.), position within the region of interest and the scale (this scale is not the same as the scale used at the detection stage, though these two scales are multiplied). For example, in case of the third line feature (2c) the response is calculated as the difference between the sum of image pixels under the rectangle covering the whole feature (including the two white stripes and the black stripe in the middle) and the sum of the image pixels under the black stripe multiplied by 3 in order to compensate for the differences in the size of areas. The sums of pixel values over rectangular regions are calculated rapidly using integral images.

The presence of a Haar feature is determined by subtracting the average dark-region pixel value from the average light-region pixel value. If the difference is above a threshold (set during learning), that feature is said to be present.

To determine the presence or absence of hundreds of Haar features at every image location and at several scales efficiently, Viola and Jones used a technique called an Integral image. In general, "integrating" means adding small units together. In this case, the small units are pixel values. The integral value for each Pixel is the sum of all the pixels above it and to its left. Starting at the top left and traversing to the right and down, the entire image can be integrated with a few integer operations per pixel.

As Figure 19 1 shows, after integration, the value at each pixel location, (x,y) contains the sum of all pixel values within a rectangular region that has one corner at the top left of the image and the other at location (x,y). To find the average pixel value in this rectangle, you'd only need to divide the value at (x,y) by the rectangle's area.
Figure 19. The Integral image trick. After integrating the pixel at \((x,y)\) contains the sum of all pixel values in the shaded rectangle. The sum of pixel values in rectangle D is \((x_4,y_4) - (x_2,y_2) - (x_3,y_3) + (x_1,y_1)\).

But what if you want to know the summed values for some other rectangle, one that doesn't have one corner at the upper left of the image? Figure 19 2 shows the solution to that problem. Suppose you want the summed values in D. You can think of that as being the sum of pixel values in the combined rectangle, \(A+B+C+D\), minus the sums in rectangles \(A+B\) and \(A+C\), plus the sum of pixel values in A. In other words

\[
D = A+B+C+D - (A+B) - (A+C) + A
\]

Conveniently, \(A+B+C+D\) is the integral image's value at location 4, \(A+B\) is the value at location 2, \(A+C\) is the value at location 3, and \(A\) is the value at location 1. So, with an integral image, you can find the sum of pixel values for any rectangle in the original image with just three integer operations:

\[(x_4,y_4) - (x_9,y_9) - (x_3,y_3) + (x_1,y_1)\].

To select the specific Haar features to use and to set threshold levels, Viola and Jones use a machine-learning method called AdaBoost. AdaBoost combines many "weak" classifiers to create one "strong classifier". Weak" here means the classifier only gets the right answer a little more often than random guessing would. That's not very good. But if you had a whole lot of these weak classifiers and each one "pushed" the final answer a little bit in the right direction, you'd have a strong, combined force for arriving at the correct solution. AdaBoost selects a set of weak classifiers to combine and assigns a weight to each. This weighted combination is the strong classifier.

Viola and Jones combined weak classifiers as a filter chain, shown in Figure 20, that's especially efficient for classifying image regions. Each filter is a weak classifier consisting of one Haar feature. The threshold for each filter is set low enough that it passes all, or nearly all, face examples in the training set. (The training set is a large database of faces, maybe a thousand or so.) During use, if any one of these filters fails to pass an image region, that
region is immediately classified as "Not Face." When a filter passes an image region, it goes to the next filter in the chain. Image regions that pass through all filters in the chain are classified as "Face." Viola and Jones dubbed this filtering chain a cascade.

The order of filters in the cascade is determined by weights that AdaBoost assigns. The more heavily weighted filters come first to eliminate non-face image regions as quickly as possible. Figure 21 shows the first two features from the original Viola-Jones cascade superimposed on my face. The first one keys off the cheek area being lighter than the eye region. The second uses the fact that the bridge of the nose is lighter than the eyes.

(* Taken and adapted from the OpenCV documentation and Hewitt, Robin, Finding Faces in Images, Servo February, 2007)
APPENDIX B: Object Tracking through MeanShift

Moving objects are characterized by their color-histograms. Therefore the key operation of the object tracking algorithm is histogram estimation. Mean-shift tracking algorithm is an iterative scheme based on comparing the histogram of the original object in the current image frame and histogram of candidate regions in the next image frame. The aim is to maximize the correlation between two histograms.

Object tracking for an image frame is performed by a combination of histogram extraction, weight computation and derivation of new location.

Figure 22. MeanShift at work on face tracking

Mean shift, a simple iterative procedure that shifts each data point to the average of data points in its neighborhood, is generalized and analyzed first in the paper by Comaniciu et al. (Real Time Tracking of Non-Rigid Objects using Mean Shift). This generalization makes some k-means like clustering algorithms its special cases. It is shown that mean shift is a mode-seeking process on a surface constructed with a “shadow” kernel. For Gaussian kernels, mean shift is a gradient mapping. Convergence is studied for mean shift iterations. Cluster analysis is treated as a deterministic problem of finding a fixed point of mean shift that characterizes the data. Applications in clustering and Hough transform are demonstrated. Mean shift is also considered as an evolutionary strategy that performs multistart global optimization.

The central computational module is based on the mean shift iterations and finds the most probable target model (its color distribution) and the target candidates is expressed by a metric derived from the Bhattacharyya coefficient. The theoretical analysis of the approach shows that it related to the Bayesian framework while providing a practical, fast and efficient
solution. The capability of the tracker to handle in real-time partial occlusions, significant clutter, and target scale variations is demonstrated in the paper.

The multivariate kernel density estimate is:

$$\hat{f}(x) = \frac{1}{nh^d} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

Where $x = a$ point,
$n = number$ of points,
$K = selected$ kernel,
$d = dimension$ of the space, and
$h = window$ radius or bandwidth.

The kernel is Epanechnikov kernel, and its formula is

$$K_E(x) = \begin{cases} 
\frac{1}{2} C_d^{-1} (d + 2)(1 - \|x\|^2), & if \|x\| < 1 \\
0, & otherwise 
\end{cases}$$

Where $C = the$ volume of the unit $d$-dimensional sphere.

The colour distribution of the object is converted to 1D distributions. The multivariate density estimate is used to weight colors based on their appearance inside the kernel. A Bhattacharyya coefficient is used to evaluate the difference $\delta$ between two distributions:

$$\delta(y) = \sqrt{1 - \sum_{n=1}^{m} \sqrt{p(y)q}}$$

Where $q = target$ model,
$p = calculated$ distribution and,
y = location.

The tracking starts initialization from which the target histogram is set. In the next frame, the surroundings of the former localization is sought to find a position in which the difference $\delta$ is smaller than the threshold.
Figure 23. Results sequence shown in the original MeanShift paper