Multiview Video Streaming
An Integrated Video Monitoring System

Advisor: Athina Markopoulou

Team: Andrew Im
Huy Vo
Sevag Kesheseh Avanessian

Table of Content:

1. Introduction:

The end of the twentieth century was characterized by the exponential growth of the computing power that all empowered disciplines of engineering further. One of the exciting developments in the field of digital communications is the idea of machine vision – to be able to use visual data from a camera to perform automated and intelligent tasks. The availability of server option would also allow for effective communication to a client that is interested in intelligent machine vision through the use of web based communications.

The ultimate goal of this project was the development of platform that would combine image processing from multiple cameras with easy access to the desired data through a server, all performed in real time. This real time multiple-camera system would be able to provide a variety of viewing options for the client. Expandability of the developed platform was also another major goal, meaning that the number of connected cameras to the system could be changed without major modification of the code, while the server side could also be altered to allow for many more clients to access the data by expanding to a commercial grade server.

As the design goals were being considered, the team decided on several major functions for the system based on what the client would need: first, to be able to switch between the different cameras connected to the system; second, to be able to activate a camera sweep function which would display all connected cameras one by one with a predetermined delay; and third, an object track process where, based on a facial detection algorithm, the camera that best displays an objects face is selected automatically.

This system, while being straightforward in its functional goals, is an interesting combination of numerous components. The various considerations for such a system ranged from choosing appropriate cameras, to methods of manipulating the video stream prior to processing, actual processing of the frames including face detection and the algorithms used to select the best camera, methods of streaming the video to the output, as well as the page design and video implementation on the server side.

2. Design Process:

The design process included dividing the design goals into major, distinguishable parts and approaching the challenges that each posed.

The first part of the system required the information to be passed to the processing algorithm/face recognition process as well as the server. The streaming would be done through software selected to fit the requirements of the project. The input stream from each view point also needs to be split to feed both the server and the image processing. Some of the considerations for this goal were using Microsoft Directshow filters to split the stream and feed it to the pre-processing while being able to provide the stream to other necessary components. Another approach was using small third-party software that splits the stream from a camera. The final solution was to use multiple cameras at one location to provide the feeds.

The second part of the system involved the server side retrieving the information relating to the camera selection from the processing stage, and displaying the cameras on

a webpage interface. This party would require availability of multiple functions on this web page, including switching between cameras, camera sweeping, as well as displaying the correct camera if the object tracking option is selected. This design would have to robust for several clients.

The final part of the project was the automated feature. In this part information from the camera is fed to a process created using the Open Computer Vision library package. This process, written in C++, serves two major functions. First, the software is able to do face detection which detects a face in the frame based on predetermined parameters and draws an estimated circumference around the facial area. Certain parameters of this circumference such as radius, relative size, and placement in the screen are used to complete the next task of the process which is to choose the best camera to see the objects face.

The design would come together as an image processing system with a server that displays the correct streams (Figure 1).

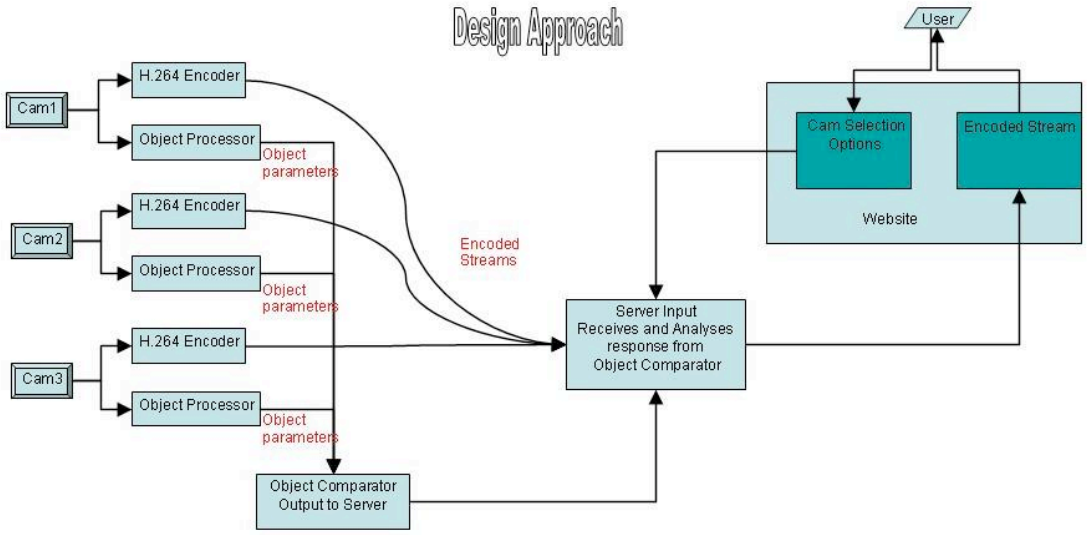

Figure 1: The design approach to the Multiview Video Streaming

3. Details of Design:

3.1) Streaming/splitting: Other than acquiring the cameras themselves, the first step of connecting them to a real time system is how to deliver the stream. The streaming challenge was choosing the appropriate software to broadcast the stream to the server, as well as splitting the stream so that the image processing can perform the task of selecting the appropriate view.

The software for streaming the data from the camera had to be selected to fulfill certain criteria. One of the considerations, of course, was that it should not consume much of the resources specially considering the demonstration was running off a laptop. Another consideration for the software was the delay time it introduced to the output. A

third issue that tied in with the considerations for delay was the capability to work with multiple cameras at the time. A suitable software would fulfill all these requirements.

The second task, as stated, was splitting the stream so it could be also feed the image processing. To accomplish this task three alternatives were considered. The first method was utilizing Directshow filters. The second approach would be using third-party splitting software to split the stream. The final method would be to physically separate the stream and feed the server and image processing independently.

The first method of accomplishing the splitting task was based on utilizing Directshow filters. Directshow is a multimedia framework and API (Application Programming Interface) introduced by Microsoft. The main function of directshow is to allow software developers access to a tool to perform various operation with media files or streams while dealing with libraries developed for a simpler interface.

In directshow, media operations are performed by blocks known as filters. There are three main types of filters: source filters, transform filters, and rendering filter. The source filter is defined as a block that takes data from a source (in this case the webcams) and introduces it into the filter graph environment. The function of the transform filter is to process data from some sort of a source filter and pass it along. Finally, the rendering filters are used to render data to any location that accepts media input (such as the screen). To manipulate the directshow filters for building functions as well as testing them, a tool called GraphEdit is used (Figure 2). GraphEdit is a tool for building filter graphs. Not only does this tool display all available filters but also makes simulation for testing different filter graphs very convenient. This tool allows for exploring simple

design concepts as well as more complicated builds that include many types of filters. (Figure 3) Unfortunately directshow filters were not ultimately used to split the stream as the intelligent switching process was not able to be characterized as a transform nor a rendering filter.

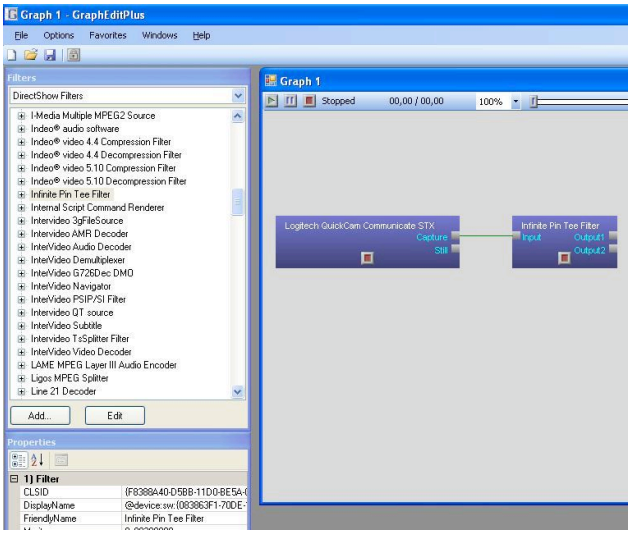

Figure 2: A simple tee filter splitting the input from a webcam

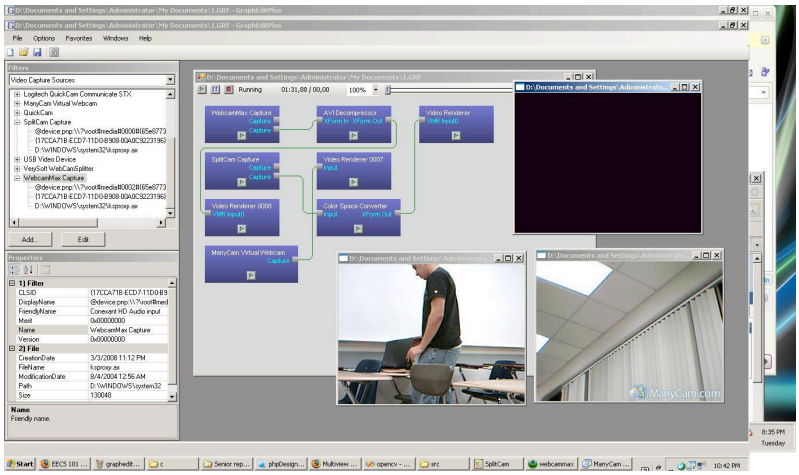

Figure 3: an example of how a variety of filters are used to process media streams

The alternative to using directshow filters was using third-party software that split the stream. (Figure 4) Multiple freeware that perform such a task for webcams were

selected. Unfortunately, the image processing was not able to accept the output from such a stream splitter as an acceptable input from a camera so this approach was abandoned as well.

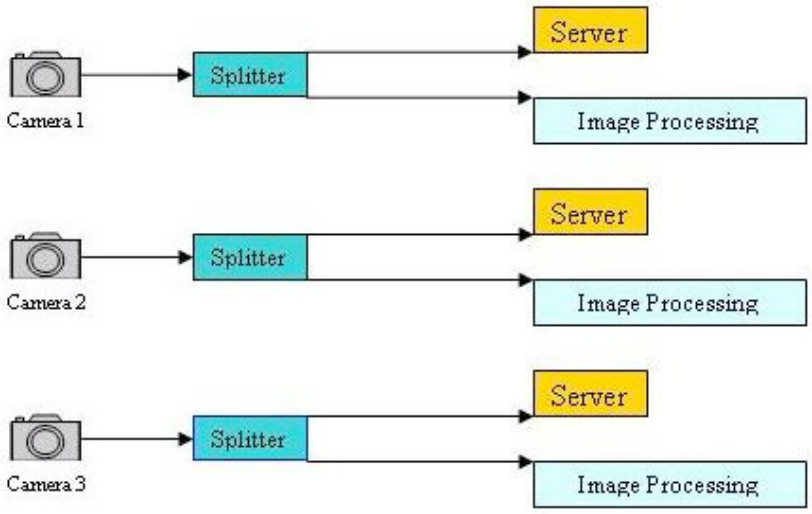

Figure 4: Illustration of how splitting software would be implemented

Finally, to allow the system to function as intended, the approach of physically separating the streams was selected. Two cameras would be supplied at each view point to enable the sever and the image processing to have access to the data stream simultaneous so the automated function of the system would be operational in real time.

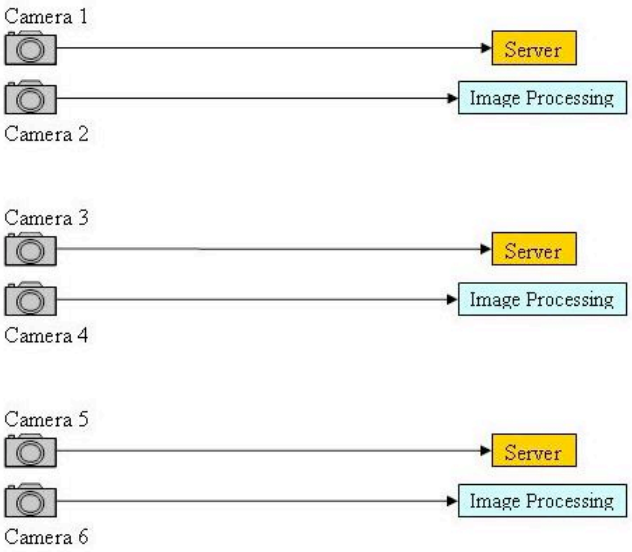

Figure 5: Final resolution to physically split the streams

3.2) Server: The objectives of the server side of this project were to serve the functions of camera selection, view sweeping, and intelligent switching (object tracking). The methods utilized to accomplish this include Javascripts, AJAX, and PhP through using an Apache server.

The initial approach to designing the server was accepting user requests through the HTTP GET method (Figure 6). A PHP script would handle the requests and the correct stream would be returned to the client. The initial website was designed with three frames. The first frame would keep refreshing every 3 seconds to see if the camera frame required changing. Frame two would accept user requests through use of forms. The third frame was used to display the appropriate video stream.

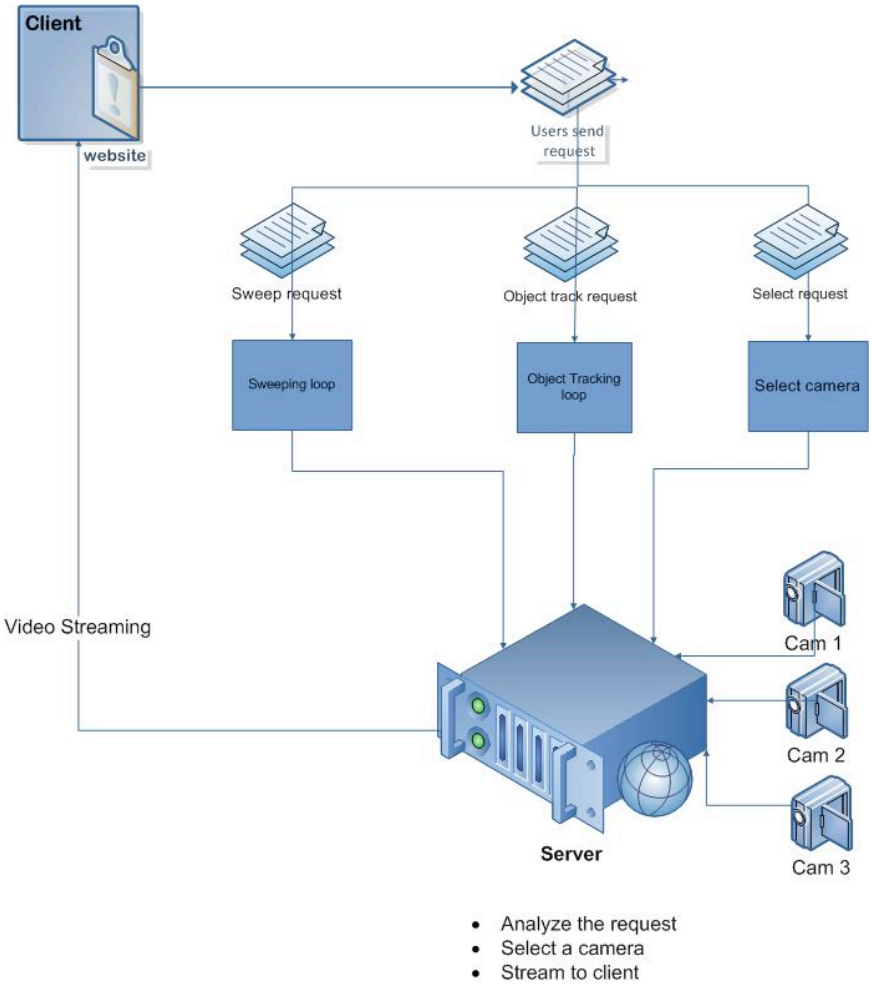

Figure 6: Initial Server

While this initial design did function, it did demonstrate some unwanted characteristics. (Figure 7) On the positive side, this server plan was easy to implement and consumed very few server resources. On the other hand, this sort of design has proven to be hard to expand, the refreshing time for the frame is fixed, it uses an uncomfortable user-interface, and the design is not very stable.

Figure 7: Initial webpage interface

To resolve these concerns about the server, and second design was implemented. This time the server uses AJAX to accomplish the required task. (Figure 8) AJAX is a web development technique used to create a web application. The new design allowed the client to enjoy increased responsiveness and interactivity with the web pages. Most importantly, the webpage now is not required to be refreshed each time there is a data fetch to the server.

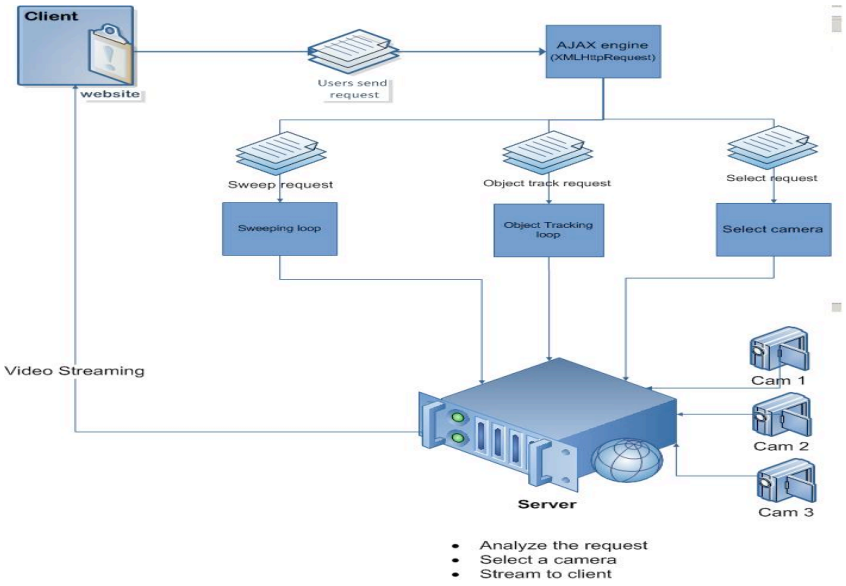

Figure 8: New server implementation using AJAX

The other important part of the operation of this server is what functions are used in the design and how they are called to accomplished the desired tasks the server has to perform. The very top of the function design is based on the user request which can choose any of the three options of camera sweeping, object tracking, and camera selection.

If camera sweeping a selected the function sweeploop() is selected. This is a function that keeps calling itself. What this function does is to call another function called camsweeping(). The function Camsweeping() takes the current camera position, move to the next camera position, display the new position by sending the appropriate stream, and finally to update the current camera position. The delay time of the function setTimeout(), which is what determines how long the loop waits to recall itself is used to set the refresh rate.

The second option, object tracking, works is a similar fashion to the sweeping function. A loop name trackloop() is called, which is another function that calls itself

using the function setTimeout(). This function, however, calls a different function named camTracking(). camTracking() reads from the file that is the ouput from the image processing, which is what is used to choose the appropriate camera to track the object. This value of the best view point is the compared with the current camera view. If the new camera view is the same as the view that is already being displayed the function does nothing. If the value for the new camera view is not the same as the one being displayed the appropriate stream is selected and sent to the client.

The final function of the server is to perform camera selection that is requested by the user. This task is accomplished using a function called get(). When get() is called the loop that may have been used in the previous instances by the user being interested in object tracking or camera sweeping is terminated using the function clearTimeout(). The user's request is passed as a string to the selectcam.php which handles the selection by analyzing the user request. If the current request is the same as what is being displayed the function does nothing. If the current request is different from the currently displayed view, the appropriate camera stream is selected and sent to the user. The current camera value is now updated to complete this function.

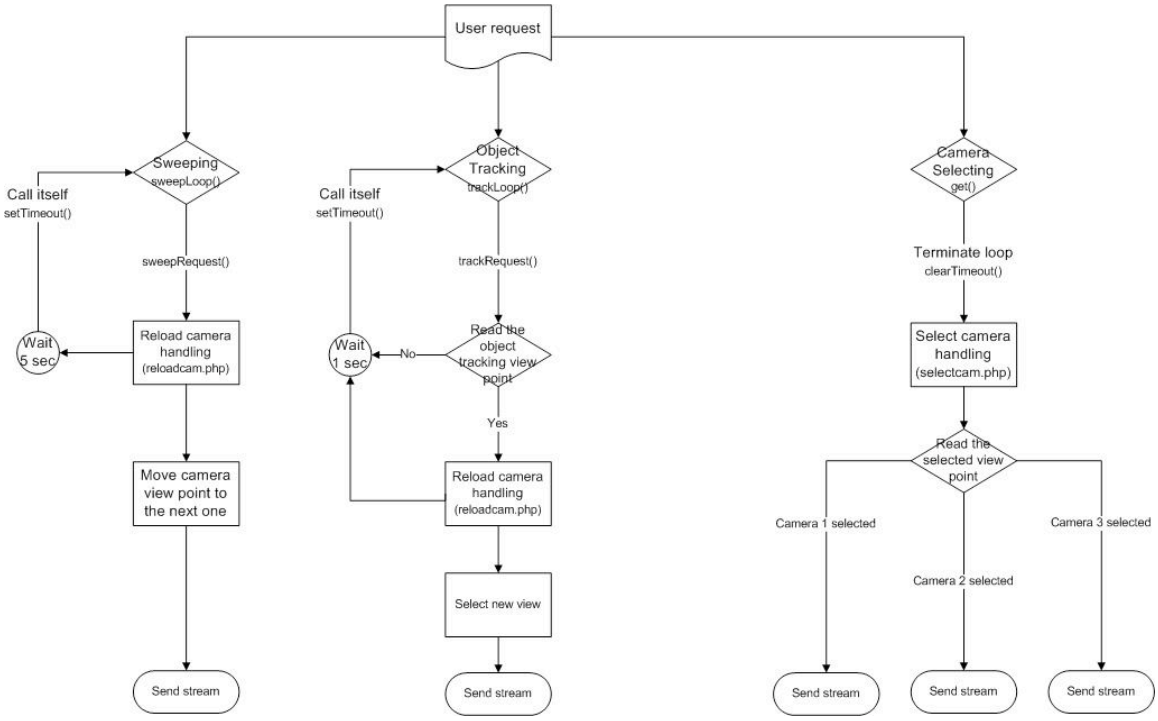

Figure 9: Chart of the function system of the server

3.3) Automated Feature: The function of the automated feature in this project is to perform the image processing for the option of object tracking. Object tracking is basically the option that bridges the gap between camera selection and camera sweeping. When the object of interest is in the viewing range of the multiple cameras connected to this system, the camera that presents the viewpoint with the best frontal facial view of an object will be selected.

Selecting the appropriate camera was one of the main challenges of this project. To solve this challenge, the research suggested using the OpenCV package. The well-known Open Computer Vision Library was originally developed by intel. The main aim of this library is real-time image processing, which would the function needed to complete the object tracking.

The decision was made to use the OpenCV face recognition algorithms to select specific data extracted from the facial parameters and store them. A comparison algorithm would them compare each of these weighted parameters to perform the camera selection. Finally the selection information would be passed along by being written to a file. This file would be accessed later by the server to perform the object tracking function.

The first part of the image processing was implementing the face recognition. To do this the existing computer vision libraries from OpenCV were used. The face recognition process in OpenCv is carried out through the Haar Classifiers. The Haar classifiers use the Haar cascade to perform object recognition – in this case trained for detecting the frontal view appearance of a face. The Haar cassifiers could be trained for other parameters such as profile view, best full figured view, etc.

Using the parameters from the preexisting vision libraries a circle is drawn in a region decided by the face recognition process. Certain parameters extracted from this detection circle will allow for intelligent switching later on. The performance of the image processing at this point is decent as each detection occurs as under 200ms per frame. This low detection time will allow for reasonable delay in the system.

In the next step, an algorithm needed to be developed so the automated camera selection can choose the face that is the best frontal view. To do this the correct parameters needed to be extracted from the face recognition process. The circle which the face recognition function draws around the appropriate region passes around its parameters such as position and area. The desired parameters used for comparison then

are the radius of the circle, its area size relative to the size of the window, and the position of the circles relative to the edges of the windows (x & y coordinate values). These values were used as weighed parameters to calculate a camera selection value that is needed next to perform the correct camera selection. The following figure illustrates how the weighed parameters would contribute to the camera selection.

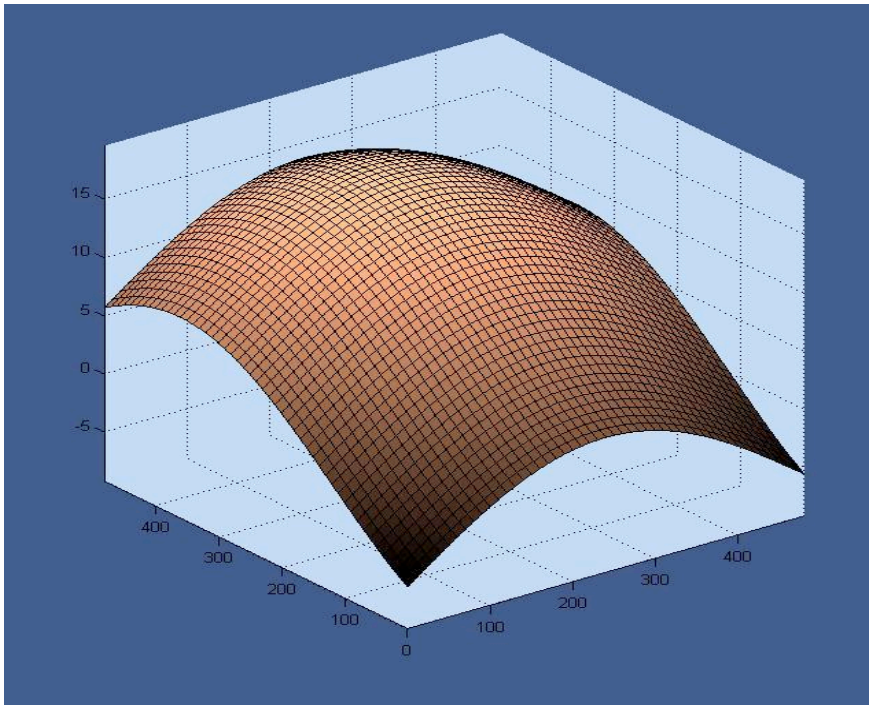

Figure 10: Algorithm that assigns weighs to the parameters based on screen location

The comparison loop in the image processing allows for making a camera selection. The conditions of the loop compare the value assigned to each camera from the previous state each time there is a detection. The comparison loop then chooses a camera and passes the appropriate value to the output. The camera(s) that do not detect a face and hence pass on zeros are eliminated from the selection process. The output file that the camera selection process writes to is read by the server to perform the camera selection task.

4. Problems and Resolutions:

The most major problem encountered in the project was overcoming the stream splitting challenge. Initially, it was assumed that the directshow filter manipulation would allow for directly splitting the stream. This proved to be not possible as the OpenCV process was not available in the filter graph environment as neither a transform filter nor a rendering filter. The third-party freeware that split the stream from the webcams also did not serve the purpose at the end. While using this method the stream was split successfully, OpenCV was not able to recognize this input either so this approach was abandoned as well. Finally, the stream was split physically by using two cameras at each view point, enabling the image processing and displaying of the desired view point to be possible simultaneously.

Another problem that required reproach was the initial immature webpage design. Using the HTTP get method required the webpage to be refreshed every time the server checked to see if a change in the displayed stream is necessary. This resulted in an uncomfortable environment for the user as the unstable design forced the webpage to reset multiple times unnecessarily. To resolve this issue, the webpage was redesigned using AJAX. The most obvious advantage of using this approach was that now the server did not require refreshing the page every time it has handling a request.

OpenCV did not pose any absolutely great problem to overcome, but was a challenge that was characterized by constant debugging and improvement of the code.

The extraction of the correct parameters was essential for detection. The weighed parameter approach also required a lot of development, as the parameters needed to handled appropriately through a comparison loop for the correct camera view to be selected.

5. Team Organization:

The team organization for the project was based on a functional approach. Since the design approach was to recognize the major components of the system and design them separately first for ease of testing, it was decided that each team member be assigned a particular component to work on. Several weeks before the final demonstration, the group concentrated on integrating the components of the system and running the final testing to grade the performance and make the necessary adjustments. Although some unavoidable overlapping occurred between the work on streaming approaches and the relevant input methods for OpenCV, the team members remained predominantly concentrated on their assigned components throughout the development of the project.

Team Member	**Class**	**Assigned Component**
Andrew Im	4th year Electrical Engineer	Automated feature, using OpenCV to evaluate the facial parameters and select the appropriate viewpoint, passing along the camera choice to the server for stream selection

Huy Vo	4th year Electrical Engineering and Computer Science	Server, retrieve the information from the automated feature and display the correct streams depending on client requests
Sevag Keshesh Avanessian	4th year Electrical Engineer	Streaming, find streaming solutions for data transport to the server and OpenCV, find splitting solution for simultaneous function of image processing and display

6. Cost Analysis:

The cost of the project was basically consisting of the cost of the webcams that were used in testing. The laptop that the server was implemented on was provided by one of the students. All of software solutions, including streaming/splitting and the usage of the Open Computer Vision Library, were based on freeware and open source material. Although the following table provides an accurate description of the cost of this particular project, the cost to a client could be varied greatly, depending on the quality and quantity of the cameras as well as the size and grade of the required server that is required for the task.

Component	**Cost**
Logitech Communicate STX (2)	$40
Creative Instant Webcam	no cost to team
Creative Instant Webcam (skype edition)	$35
Creative Live! Cam Video IM	$30
Microsoft VX1000	$30

Total Cost	**$135**

7. Outcome

As stated, the ultimate goal of the multiview streaming system was the development of a platform which successfully performs image processing as well as displaying the stream based on client request, all in real-time. Although some of the initial assumptions of this project proved to be flawed, the final outcome is an integrated one that does perform the expected functions. The expandability of the system was another aim of this project. With improved coding, or possibly a future release of the OpenCV library which would allow for easier access to a variety of inputs, and taking advantage of commercial grade servers, this system could be expanded to a great degree.