Real-Time Human Tracking through Single-Channel Video Feeds

Benjamin Martin

Abstract – In order to effectively process, locate, and track human forms real-time in a video feed requires not only a strong hardware base but adaptive and intelligent software solutions. Likewise, in order to locate objects in 3d space with a single channel video feed requires additionally intelligent software. This project strives to develop a system that performs all of these things via attention-specific systems, camera calibration techniques, and facial-bounding-box detection algorithms.

Introduction – Real-time video processing and automated surveillance has been of large interest in many fields. It is most often thought of us practical for intelligence and security, but it has many more potential applications. This document attempts to develop a practical and inexpensive solution for a somewhat "interactive" video monitoring system. This will allow, in particular, performing arts to be able to track and location of a performer on stage and program an automated response. An example of this would be a light following a dancer on stage. Traditional methods require either personnel to manually operate a spotlight – which has its own set of limitations, or for the dancer to move to pre-recorded movements. This project attempts to create a solution which will be able to follow the dancer in three dimensions, allowing for error or improvisation in the case of pre-recorded movement. This also hold as a benefit over manual lighting control, as it can operate any number of systems simultaneously and account for depth, which most manual systems lack. This document will outline the anticipated methods of research and hypothesis for the general detection of human forms.

Objectives – The objective of this project is to design and implement a system which tracks human forms on a stage. It also will be designed to be able to provide a user with the 3-dimensional coordinates of any given dancer on the stage. A major proponent of this design is to be able to interface this data with additional applications that will use the geometric coordinates of the detected people to produce artistic effects. This will require an external api to be implemented.

Hazards – Several hazards exist in this project. The first real hazard is that of camera data rates. A typical USB camera has a small lag between the time the camera's sensor detects the image and the transmission to the PC is complete. This lag can be potentially very hazardous if the performer is moving quickly. Research will have to be done prior to implementation to determine whether or not this lag will be acceptable, as when the camera is far away

from the object the small delay will have a less pronounced effect. This research will either conclude that the USB camera is acceptable, or a faster camera will have to be found. The goal of this project is to produce a result that is not only simple but very cost effective, thus an Ethernet or Firewire based camera will be acceptable, but not ideal.

The second hazard is accuracy. At this point in the design phase, and going along with the project goal of being both cheap and simple, one camera will be ideal. This presents a very technical challenge in accuracy. It will require additional calibration of the camera in order to accurately determine their position along the z-axis. While this will work fairly cleanly in theory, it may be found that in reality the accuracy just isn't there. At that point a design decision will have to be made regarding the utilization of an additional camera to produce two reference frames.

A final hazard is resolution and frame rate of the camera, as this will determine hardware requirements. For a system to be fast enough to track a performer reliably, estimated frame rates of 15 – 30 FPS will be needed. If the camera cannot this frame rate at the full resolution, lesser resolutions will need to be selected at the price of accuracy. As stated in the hardware section, potential lossy frame rates or low resolution will require that additional hardware will need to be used.

Hardware Components – The hardware design required for this project is largely dependent on the specifications of the camera used. If it tends to have low frame rates or resolution, a converter will have to be implemented or purchased in order to allow a generic composite signal from a video camera to be recognized as a webcam by the PC. This will allow for much higher frame rates and resolution, and thus much higher accuracy. A second component necessary will be an RF link between the video capture device and the PC. The preferred protocol will either be Wireless USB or Bluetooth, as both are fairly cheap to implement, range depending.

Software Components – Software development will make up the bulk of the project. Several software components will need to be implemented as follows:

Image Extraction - This component will be

Camera Calibration – This component will be the critical point in accurate depiction of the coordinates of an object. It will require, first off, the calibration of the stage. By identifying four concrete points on the stage, a depth can be determined and likewise other objects will be able to compared on the same scale.

GUI – While a majority of the low-level processing will be written in C++ for the sake of performance, the GUI will be built through ActionScript. This will allow the GUI to be built more quickly and efficiently, while not compromising much in the way of performance.

Motion Detection – Motion detection will be performed through background subtraction. The resulting regions will then have a threshold applied to eliminate any noise, and passed through for object recognition.

Object Recognition – In particular, facial recognition will be relatively complicated. At this point in the design, the chosen algorithm will be based off of a neural network that has been trained with several front and side view faces. It will then analyze the regions in which a moved object was detected, and determine if a face is contained within the region.

Object Tracking – Tracking the faces is very straightforward, as points will need to be stored at a given resolution-defined interval. Trends can then be found by finding linear regressions and approximations on the curve.

Inverse Projection – The most intensive part of the project will be to derive a method for accurately determining a given object's coordinates. While the x and y component of the object will can be easily found through inverse projection, the z-component will require a system for calibrating based on extracted head size. By utilizing a transformation matrix and two or more calibration points for a person or object, the z dimension will be able to be extracted.

API – This system is designed exclusively to work with other pieces of software to add the artistic elements to a performance. As such, an API will need to be implemented in order to communicate the findings. For scalability reasons, the API will be implemented via network interfaces. A TCP connection will be established and the client will be required to poll the existing state for locations or events.

Attention Selection – Due to the fact that extra faces will not be found within close to 80% of the visual field, a simple algorithm can be used in order to ease the processor load. By only analyzing the edges of the field and tracking formerly detected faces, a large percentage of the field will not have to be analyzed.

Workflow – The entirety of the project is expected to take approximately two and a half months. It is expected to be broken into essentially four sprints. Each sprint will entail the typical Scrum model of development and testing per sprint. They are outlined as follows.

Sprint 1: Motion Detection – This sprint will be two weeks long and set up the basic

framework for which the rest of the sprints depend on. The goal for sprint 1 is to develop an application that can accurately track human faces over time. It is also will support a frame rate over 30 FPM (leeway is needed as the inverse projection phase will also be fairly processor intensive).

Sprint 2: Inverse Projection – The goal of this sprint is to implement a theoretical method for determining the x, y, and z positions for an object given known reference points and the calibrations of the camera. This sprint will be the longest, taking upwards four weeks.

Sprint 3: Camera Calibration, Video Transmission – Camera calibration methods and research will be developed over this sprint. It will have to be integrated with the previous sprint, as both have some degree of codependence on one another. The video transmission component of this project will be the design and implementation (or purchased, budget depending) of the short-range wireless video link.

Sprint 4: Integration/API – Once the basic system is in place, the final step is integration and the API development. The API will be very straightforward and not take more than one week to construct. Integration, however, will make it evident whether or not design considerations need to be changed. This will likely take upwards one month.

The workload of this project will be completed mostly by myself, with assistance and guidance in implementation from Dr. H. James de St. Germain. Periodic checks and detailed scrum models will help to gauge progress and determine if design requirements can be pushed forward or back.

Skin Detection Hypothesis – Based on several studies performed in the area of human skin detection, it is going to be assumed that the most efficient method for isolating human forms will be to use color modeling of human skin. It is apparent from these that regardless of race, human skin can most easily be represented by the HSV color scheme, which isolates the hue, saturation, and intensity (value). This allows for detection in harsh lighting conditions, as the hue is irrespective of shadows or highlights. It was mentioned prior that race is not an issue with this proposed model, as it has been determined through several statistical studies that the only legitimate difference between dark skinned persons and their caucasion counterparts is the saturation of the hue.

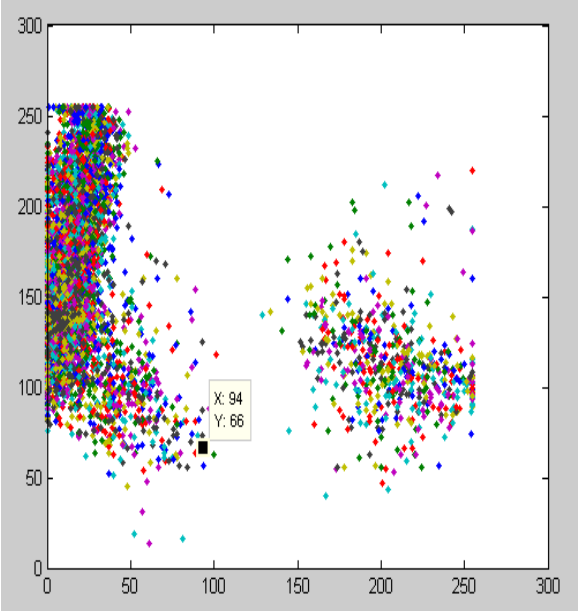


Figure 1 - It can be seen from this figure that there are two real clusters of hue (the x-axis) and saturation (the y-axis). The cluster on the left if the general range for skin tones on illuminated skin, while the right section is that of skin in a reflected shadow.

The values to be used in isolating the skin tones have been found through making a skin map of several people, and plotting the respective H and S values (we will ignore V as it only tells us how bright the image is). By plotting these respective to one another, a very narrow clustering of values can be found, and as such threshold values can be determined. Upon thresholding the image, basic histogram analysis may be used in isolating the eye locations.

Histogram Analysis – The histogram analysis portion uses a basic heuristic that will be developed to cheaply isolate the eyes given a general face mask grayscale image. This is possible due to the face that the eyes are always the darkest point on the face (with the exception of similarly dark points at the nostrils). By inverting and equalizing the image and applying a dynamic thresholding operator on the image, several distinct points emerge. At this point a histogram is generated in both the x and the y direction. Since the eyes are brighter than the surrounding skin regions, it is known that there will be a relative peak in the histogram. By analyzing the image strips at each peak, we can perform some basic operations that will allow us to locate the exact point of the eye.

Eye Detection – Upon generation of the image strips, another horizontal histogram is created. Since it is known that there will be two bright points surrounded by dark space on either side and in the middle, the peak that is associated with the eyes is used to compute the basic bounds of the eye. From this point, if low accuracy is desired these bounds may be used. Otherwise, the peak at which the brightest point occurs is most likely the center of the pupil.

Validation – This method, while very cheap, shows many false positives (eyes detected where they do not exist). The anticipated method to remedy this is to perform several sanity checks on the list of potential eye points. The first is the use of the center of mass. By finding the center of mass of the

skin mask, the general center of the face will be found (though this value will be slightly low due to the presence of the neck or other limbs in the frame). By using general proportions of human features, several of these values may be eliminated. Such conditions as the angle between the eyes may not be over sixty degrees, the eyes must be approximately the same size, and that they are within 25% of the head's height from the center are a few that will be used. Preliminary tests show that these conditions eliminate most false positives.

Object Tracking – Once a lock on a face has been established, a method needs to be utilized to allow the face to be tracked in real time. To do so requires an algorithm to be implemented that will a very small portion of the CPU. The algorithm we will use will be a derivation of the Mean Shift algorithm, as proposed by Intel in 1998. The implementation and description of this algorithm will be highlighted in the following section.

Though object tracking is not the original purpose of the mean shift algorithm, in color video streams it proves a very computationally inexpensive method for tracking relatively solid-colored objects. The point at which this algorithm fails is that it can really only track objects of a consistent size. In a video stream with live objects, however, distance from the camera is directly proportional to the perceived size of the object. With this in mind, it is clear that an object that changes its position along the z-axis will appear to change in size. This requires us to extend the mean shift algorithm as proposed by Intel. The extension relies on continually adapting size of the search window, allowing the position along the z-axis to not only be mitigated, but to be calculated and used in control applications.

Continuously Adaptive Mean Shift - As was aforementioned, the continuously adaptive mean shift algorithm revolves around color histogram analysis. The first step, then is clearly to define the appropriate ranges of color.

As was stated in the section regarding human skin tones in the HSV color spectrum, a very narrow band can be isolated and utilized as a sort of background subtraction, eliminating the necessity for additional background subtraction.

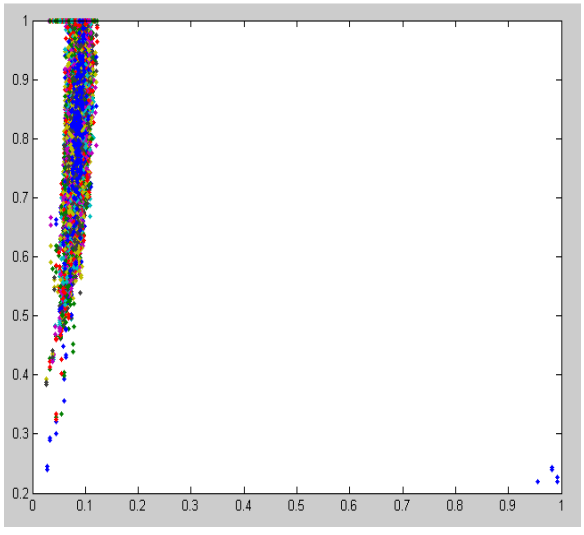


Figure – This shows the narrow band of hue and saturation for a given skinmap. The X axis

represents the hue, while the Y axis represents saturation

It is noticed that there is a small cluster of points on the rightmost side of the graph. This is due to the fact that the hue band is a circular measurement of degrees. That is, the largest hue is really immediately adjacent to the smallest potential hue. The narrow window of hue and saturation is quite resilient against both light intensity and skin tones.

Converting RGB to HSV

$$V = \max(Red, Green, Blue)$$

$$\Delta = \max(Red, Green, Blue) - \min(Red, Green, Blue)$$

$$S = \frac{\Delta}{\max(Red, Green, Blue)}$$

If (Red == V) $H = 60 * (\frac{Blue - Green}{\Delta})$

If (Blue == V) $H = 60 * (2 + \frac{Red - Blue}{\Delta})$

If (Green == V) $H = 60 * (4 + \frac{Red - Green}{\Delta})$

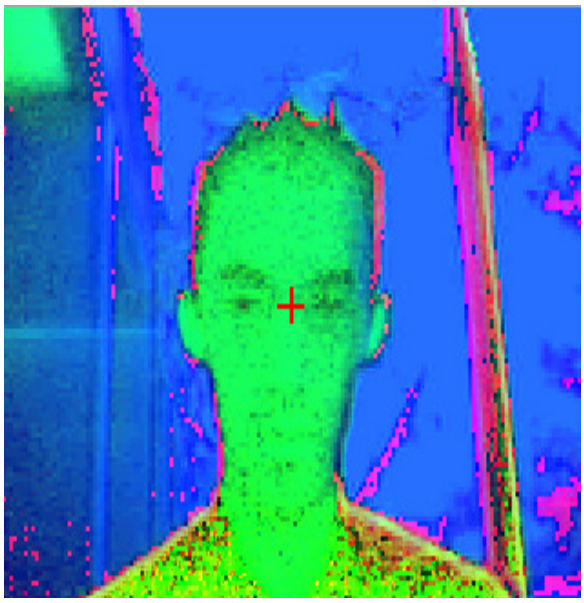
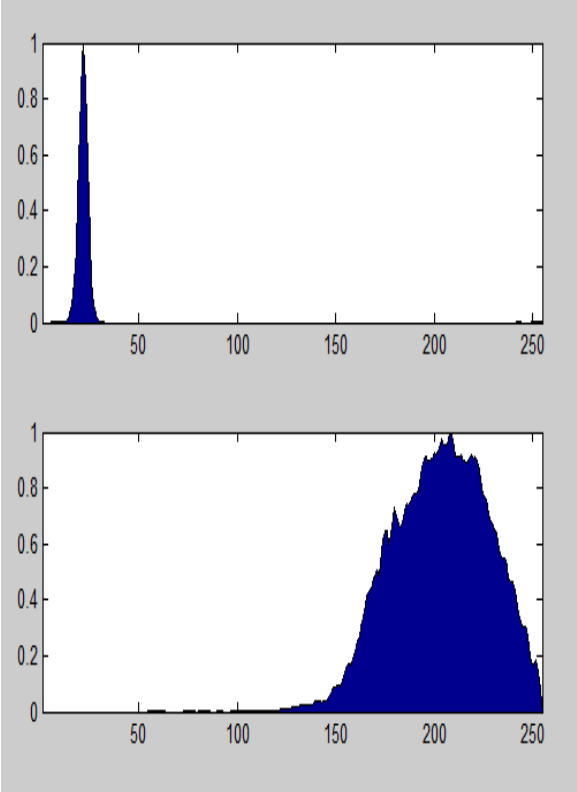


Figure – This shows the HSV values for a typical image sampled by a webcam. It can be seen that the human skin tones provide a sharp contrast from the other regions

Once an image is converted to its respective HSV color, an initial histogram must be established so that the unrelated regions of the image may be subtracted, By specifying a probability distribution for each individual pixel, large clusters of appropriately colored images can be extracted. The following figure shows the histogram distribution of hue (which contains a very narrow window), and saturation, which covers a much broader base. By using these two paramaters, a very reliable skin base can be detected.



The skinmap image can be defined by the following equation:

$$P_{robability\ Image} = \sum H(i,j) * S(i,j)$$

By iterating over each pixel and using the histograms as lookup tables, the product of the hue and the saturation values are placed into the corresponding locations. The resulting image is a grayscale skinmap that is used for the remainder of the algorithm.

In our testing implementation, the user is prompted for a color selection window as an alternative to the traditional skin tones. In order to use this histogram, a simple filter is applied. This allows the continuously adaptive mean shift algorithm to track any uniquely colored object by generating on-the-fly probability distribution histograms.

Upon converting the HSV colormap image to the skinmap, an initial search window needs to be identified. This will allow the search window to scale the probability distribution and resonate over the actual bounds of the image. The initial window, due to the fact that the algorithm scales and resizes, does not need to contain the object. While selecting the actual object increases the speed of the initial mean shift, the efficiency of this algorithm is such that on a 1.2 Ghz laptop with an XNA implementation, the scaling of the window to encapsulate the entire face took less than 0.3 seconds.

The actual mean shift operation is a repeated sequence of finding the centroid of the search window, repositioning the search window around the centroid, and then rescaling the search window according to the sum of the pixels within the search window. By continually repeating this operation until the search window stabilizes around a particular location, the center of the blob can be tracked and followed very efficiently (in our test application 30 frames per second were easily achieved).

Actually determining how to resize the window is typically very dependent on the object being tracked. Due to the fact that we are targeting this application chiefly to human faces, it can reasonably be assumed that the objects being tracked will be

somewhat oblong. It is also assumed that the faces are not tilted very far sideways, and so a basic window can be set up using the following formula:

$$s = 2\sqrt{\frac{M_{00}}{GrayValues + 1}}$$

With this in mind, our implementation converted the image to 8-bit grayscale values. Our denominator, then, was 2^8 + 1, or 256. By also assuming that the windows is taller than it is wide, the actual search bounds are set by claiming s as the horizontal edge of the next search window. The height, then, is experimentally determined to work best at about 1.2 * s.

One of the largest benefits of this algorithm is its near-immunity to noise and similar objects within the scene. Upon correctly sampling the skin colors, any deviations in noise or lighting generally do not impair the results dramatically. If other objects come in or out of the scene, as well, the target is still generally maintained. The only real exception to this is if a similar sized and colored blob overlaps the target's face. If this happens, the target is occasionally "passed off", and the new target becomes the overlapping blob. This took a considerable amount of effort to accomplish, however, so the likelihood of it happening accidentally in a typically scene is minimal.

Aside from the proposed uses for this technology, there are countless others. The concept of using facial recognition and using this as a control interface for a given system is monumental. The concept used here, for instance, was implemented in an XNA game. The game used facial tracking to control a spacecraft that moved around a galaxy in attempt to reach a landing platform. By properly mapping the appropriate controls to the head position in the webcam, a pitch, yaw, and roll controller was implemented that could maintain respectively high frame rates on a laptop running at a mere 1.2 Ghz. This technology could also be widely used for those with accessibility needs. For example, by tracking the pupils of a person and detecting a certain blink pattern as clicks, the user could control the mouse just be looking at the appropriate control on the screen.

Risks – The largest risk for this project revolves around the prospect of the cpu not being able to handle the large number of operations required to analyze and process each frame. As this is not easily remedied given the budget and project specifications, the only alternative is to use very efficient algorithms and eliminate any unnecessary operations. One particular concept which will be referred to as estimated vector projection will be used. This system will monitor a history of the movement of the eyes of the subject. By doing so, the

system will be able to analyze a given trajectory and reasonably estimate where the eye will be in the next frame (given that the eye will not move out of but a small window given a 30 fps feed). By doing so, only a small region will have to be examined for eyes. That being said, this operation will still require a "lock" to be acquired, but once done will be incredibly efficient as the image size will be reduced from 320x240 to something more in the neighborhood of 50x16. It is certain that even the most basic of mainstream cpu's can operate on an image this size in real-time.

An additional method for processor offloading revolves around the fact that a majority of the processing power will be devoted to drive a GUI to help monitor and debug the system. The GUI will also be responsible for capturing webcam data, as C++ does not have any inherently clean methods for doing so. Because of this, a client-server application will be developed. This will also allow the operating system to utilize several cores for the image processing, further increasing the potential frame rates.

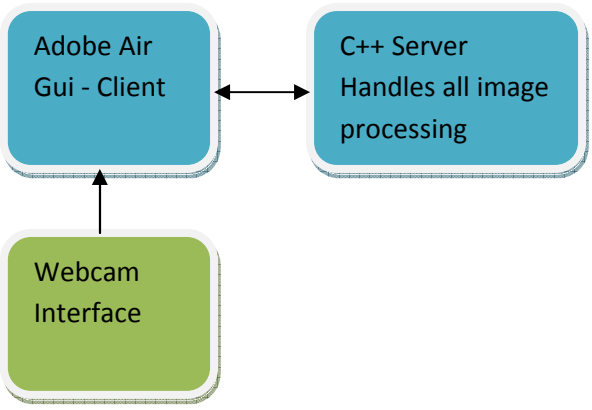


Figure 2 – This flowchart shows the relationship between the software components to be used in the project. The blue boxes represent custom software, while the green is an operating system component. These components operate through an internal network connection

If in the case that accuracy is a problem and the proposed algorithm fails, a basic eye searching window will be able to be drawn upon a successful lock and more complicated and accurate techniques such as haar object recognition or template matching may be used to extract the eye locations.

One particular hazard that we encountered was due to the face that the project we were using for overhead illumination did not create a very large field, meaning the dancer had something in the neighborhood of a 36 square foot region to dance in. This was unacceptable, so the right-angle projector box that was used had to be tilted so that the field was elongated. This had to be corrected for in the code – the implications being that we had to re-stretch the camera's view when detecting such things as the positions of the dancers. In an ideal situation, a projector with a wider field would be used. Due to time constraints for the final performance, however, a new projector was not able to be obtained.

Bill of Materials – The materials required for this research project are fairly minimal.

All that will be needed is a webcam that is capable of generating 30 fps at 320x240 resolution. Additionally, a CPU with development environments including Cygwin and Adobe Air will be used. For the actual performance, several other components will be needed. USB extension cables will be required (at about 50 feet per cable), a VGA extension cable, two projectors with wide lenses, and two webcams. Additionally, a mirror box will be used for the downward projection.

Conclusion – Upon completion of this project, a lot of research can be done in terms of scalable systems to be used in not only the Fine Arts industry, but in the general computing and in particular the gaming industry. It will allow for a new generation of lighting effects, as well as be extensible to other fields and industries.