

## **II. Analysis**

The following section contains all project related analysis documents for the Silhouette project. This includes a list of supported activities, a detailed description of the human-computer interface, and other important information such as ethical issues and security/fault requirements.

### **II.A.1 General Functionality Overview**

The overall goal of the Silhouette project is to successfully implement a working application which accepts an image stream from a live network-camera, and applies a series of algorithms to the image aimed at identifying obvious shapes and colors. Additionally, users may configure the application to identify various shapes by using a virtual drawing pen to “scribble” shapes on a mini-canvas. Silhouette will attempt to recognize the drawn shapes, and if successful, will compare those recognized shapes to the real-time camera stream.

While this may seem like a simple task, it is actually more difficult than one may expect. For example, the human mind is clearly capable of determining the difference between an apple and banana, or a cardboard box and a pool-cue. However, training a computer to recognize shapes is a daunting task, and has been studied by many computer engineers since the birth of computer graphics and “machine vision.”

## II.A.2 An Abstract Problem

The Silhouette project itself consists of two main problems:

1. **Real-time Edge Detection** - An algorithm will be used to detect the "Edges" within a real-time streaming color image. Once all edges in the image have been fully detected, it can be scanned for shapes using various image recognition algorithms.
2. **Real-Time Image Shape Recognition** - Identifying planar shapes on a white canvas is a fairly straightforward task with the correct algorithm. However, identifying shapes using a powerful and accurate edge detection algorithm from a real-time network stream, is another challenging problem.

To successfully address these problems, I will approach each one a bit differently. First, real-time edge detection will be implemented using a popular edge detection algorithm known as the Sobel Edge Detection Algorithm. The Sobel algorithm uses color gradients to determine the intensity of an edge between a set of pixels within an digital image. Secondly, real-time image shape recognition will use a powerful edge-detection algorithm to generate obvious contours within an image. The application must then apply another algorithm to locate and identify known shapes detected by the 2D planar shape recognition system.

### **II.A.3 Supported Activities (Requirements)**

The following list provides a detailed overview of the application's Supported Activities. These activities describe the functionality which must be included in the final project deliverable.

1. Users can define and configure a given network-camera JPG stream.
2. Users can adjust the algorithm tolerance for which to compare shapes.
3. Users can adjust the real-time image stream refresh rate.
4. Users can select shapes from a window, which indicates to the application which shape(s) the user would like to scan for.
5. Users can stop and start the real-time comparison engine.
6. The application applies the shape and color detection algorithm to each image captured by the network-camera.
7. The application alerts the user of any matches by superimposing a wire frame polygon around the detected shape.
8. Users can pause the real-time stream at anytime to focus on a given frame.

#### II.A.4 Human-Computer Interface Description

As aforementioned in the project proposal, the Silhouette human-computer interface will be designed using Java Swing. Ultimately, the project's interface will allow users to easily configure and access application functionality.

The interface will be designed using a popular method known as embedded internal frames. In other words, the main application frame itself will consist of multiple smaller frames which function as individual windows. For example, please see Figure II.A.4.1 and Figure II.A.4.2.

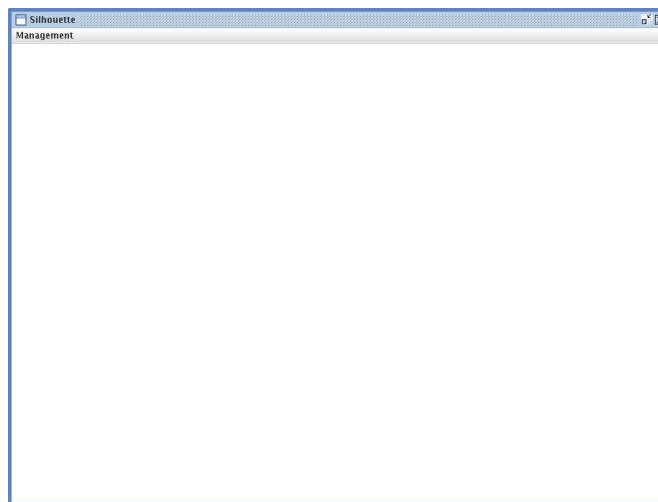


Figure II.A.4.1. When the application initially starts, the main application window opens. Any subsequent frames or window panes which open are created as individual windows within the main application window.

CMSI 402 SENIOR PROJECT LAB  
Analysis - Silhouette Project - Mark Kolich

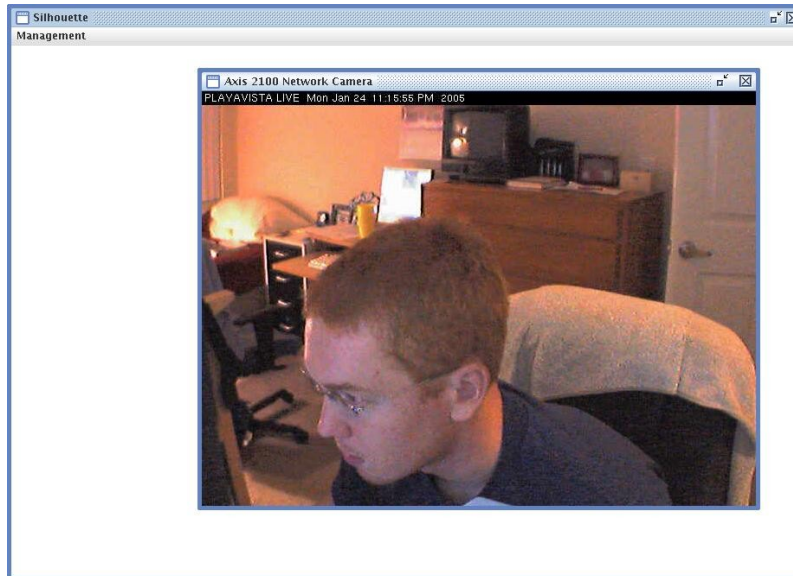


Figure II.A.4.2. Subsequent windows are opened as new internal application frames within the main application window. This system uses the popular JInternalFrame class to build the necessary interface.

Users can also control various application settings using the provided drop-down application menu at the top of the window. The application menu itself will consist of standard controls: File, Edit, Camera Management, and Help. These tools will allow users to customize the camera stream, and other important application settings.

### **II.A.5 Ethical Issues**

Any application which systemically captures and stores video from a camera can often raise serious ethical concerns. More specifically, those individuals being watched are most often concerned with their personal privacy, and usually feel restricted or even violated when recorded on camera without their consent. Therefore, it is important to note that Silhouette is not designed to provide individuals, institutions, or government agencies with a means to unlawfully invade the privacy of an individual. Additionally, those using Silhouette for educational and experimental purposes should not place a camera in a hidden location, or use the application in way which violates any local, state, or federal privacy laws.

### **II.A.6 Security/Fault-Tolerance Requirements**

As previously mentioned, capturing video data from a real-time camera can raise many serious ethical concerns. Therefore, it is extremely important that any user who is collecting potentially sensitive data has taken the necessary security precautions to maintain the stability, security, and safety of such information. The user should be aware of the following:

- 1) The user must ensure that real-time video data is not intercepted as it travels over a network from a camera to the application. Packet-sniffing is a common data-interception method used by many criminals and hackers alike.
- 2) Once the user has gathered any potentially sensitive images, they should ensure the data is protected from intruders.

## II.B.1 Use-Cases

The following use-cases are derived from Silhouette's Supported Activity (Requirements) list found in Section II.A.3 of this document.

### II.B.1.1 Req. #1 - Define and Configure a Network Camera Stream

- **Pre-Condition:** The application successfully opened and is waiting for user input.
- **MFE:** The user selects Camera, Add Camera from the drop-down JMenu at the top of the application window. A window is opened with several available text-fields: camera IP address, user name, and password. A slider is also available which allows the user to configure the refresh rate of the camera image.
- **Post-Condition:** Once all fields have been completed, the user clicks the Add Camera button. The application instantiates the camera as an available device, and adds it to a list of available cameras.

### II.B.1.2 Req. #2 - Shape Comparison Tolerance

- **Pre-Condition:** At least one camera configuration is available to the application, the user is currently viewing the live camera stream, and at least one shape on the mini-canvas has been identified by the application.
- **MFE:** Within the stream control window, move the tolerance slider to adjust the algorithm tolerance.
- **Post-Condition:** The application applies the new tolerance setting in real-time and refreshes the camera stream.

### II.B.1.3 Req. #3 - Adjusting Real-Time Image Refresh Rate

- **Pre-Condition:** At least one camera configuration is available to the application and the user is currently viewing the live camera stream.
- **MFE:** Within the stream control window, move the refresh rate slider to adjust the image refresh rate. Valid refresh rates can be as short as 10 ns, or as long as 5 seconds.
- **Post-Condition:** The application applies the new refresh rate setting in real-time to the camera configuration, and adjusts its refresh rate accordingly.

### II.B.1.4 Req. #4 - Shape Selection

- **Pre-Condition:** At least one camera configuration is available to the application and the user is currently viewing the live camera stream.
- **MFE:** From within the shape selection window, the user selects a shape from a library of known shapes: triangle, square, circle, rectangle, diamond, and oval.
- **Post-Condition:** The application initializes and begins to scan the real-time stream for the selected shape(s).

### II.B.1.5 Req. #5 - Start & Stop the Comparison Engine

- **Pre-Condition:** At least one camera configuration is available to the application, the user is currently viewing the live camera stream, and at least one shape on the mini-canvas has been identified by the application.
- **MFE:** Within the stream control window, the user clicks the "Stop" button which will pause the real-time shape location and comparison engine.
- **Post-Condition:** The application will no longer search for recognized shapes in the real-time network camera stream.

#### II.B.1.6 Req. #6 - Shape/Color Detection Engine

- **Pre-Condition:** At least one camera configuration is available to the application, the user is currently viewing the live camera stream, and at least one shape has been drawn on the mini-canvas which is recognized by the application.
- **MFE:** The application will continuously search the real-time network stream looking for recognized shapes. When a shape is found, it will superimpose a wire-frame outline of the shape on the original camera stream image.
- **Post-Condition:** The application continues to search for additional recognized shapes and colors.

#### II.B.1.7 Req. #7 - Superimpose Wire-Frame Shape

- **Pre-Condition:** At least one camera configuration is available to the application, the user is currently viewing the live camera stream, and at least one shape has been drawn on the mini-canvas which is recognized by the application.
- **MFE:** When a recognized shape is found in the real-time image stream, the application will superimpose a wire-frame outline of the shape on the original camera stream image.
- **Post-Condition:** The application continues to search for additional recognized shapes and colors.

#### II.B.1.8 Req. #8 - Pausing the Real-Time Camera Stream

- **Pre-Condition:** At least one camera configuration is available to the application, the user is currently viewing the live camera stream.
- **MFE:** The user clicks the "Pause" button in the camera control window.
- **Post-Condition:** The application pauses and will not refresh the real-time image until the user clicks the "Resume" button.

## II.B.2 Formal Scenarios

The following formal-scenarios are derived from Silhouette's list of use-cases found in Section II.B.1 of this document. Formal scenarios are designed to help better explain how a user interacts with the system and its various components.

### II.B.2.1 Req. #1 - Define and Configure a Network Camera Stream

- **Pre-Condition:** Mark has successfully opened the application.
- **User:** From the applications drop-down menu, Mark selects Camera, Add Camera. A window is opened asking for a camera IP address, a user name, and a password. Mark enters "24.130.213.11:8085" for the IP address, "mark" as the user name, and "silhouette" as the password. Mark then uses the refresh rate slider to adjust the real-time image refresh rate to 500 ns. Mark clicks the "Add Camera" button to instantiate the camera with the application.
- **Post-Condition:** The camera is now available to the application and is displayed on the available camera list.

### II.B.2.2 Req. #2 - Shape Comparison Tolerance

- **Pre-Condition:** Mark has configured at least one camera and is now viewing the real-time stream after drawing at least one recognized shape on the mini-canvas.
- **User:** In the camera control window, Mark adjusts the shape Comparison tolerance slider to 9. The application is less strict when making shape comparisons. Mark then adjusts the shape Comparison slider to 2. The application is now more strict when making shape comparisons.
- **Post-Condition:** The application continues to locate shapes as normal with its new tolerance setting.

### II.B.2.3 Req. #3 - Adjusting Real-Time Image Refresh Rate

- **Pre-Condition:** Mark has configured at least one camera and is now viewing the real-time stream.
- **User:** In the camera control window, Mark uses the refresh rate slider to 10 ns. The application now refreshes the camera-stream every 10ns. Mark now adjusts the slider to set the refresh rate to 5 seconds. The application now refreshes the camera-stream every 5 seconds.
- **Post-Condition:** The application continues to locate shapes as normal with its new refresh-rate.

### II.B.2.4 Req. #4 - Shape Selection

- **Pre-Condition:** Mark has configured at least one camera and is now viewing the real-time stream.
- **User:** Mark selects a circle from within the shape selection window. The circle is recognized, and the application initiates the real-time shape Comparison engine.
- **Post-Condition:** The application now looks for circles in the real-time network image. When found, it superimposes a wire-frame image highlighting the found shape.

#### II.B.2.5 Req. #5 - Start & Stop the Comparison Engine

- **Pre-Condition:** Mark has configured at least one camera, is viewing the real-time stream, and has drawn at least one shape on the mini-canvas.
- **User:** Mark draws several shapes in the mini-canvas window: a triangle, a circle, and a square. All three are recognized by the image library, which initiates the real-time shape Comparison engine. Mark decides he does not want the application to identify shapes from the real-time image stream. Mark clicks the "Stop" button in the stream control window.
- **Post-Condition:** The application no longer applies the shape detection algorithm to the real-time image stream until Mark clicks the "Start" button in the stream control window.

#### II.B.2.6 Req. #6 - Shape/Color Detection Engine

- **Pre-Condition:** Mark has configured at least one camera and is now viewing the real-time stream after drawing at least one recognized shape on the mini-canvas.
- **User:** Mark draws several shapes in the mini-canvas window: a triangle, a circle, and a square. All three are recognized by the image library, which initiates the real-time shape Comparison engine. The application searches for recognized shapes in the real-time camera stream.
- **Post-Condition:** The application continues to apply the shape/color detection algorithm to the image stream.

#### II.B.2.7 Req. #7 - Superimpose Wire-Frame Shape

- **Pre-Condition:** Mark has configured at least one camera and is now viewing the real-time stream after drawing at least one recognized shape on the mini-canvas.
- **User:** Mark draws several shapes in the mini-canvas window: a triangle, a circle, and a square. All three are recognized by the image library, which initiates the real-time shape Comparison engine. The application searches for recognized shapes in the real-time camera stream and superimposes a wire-frame image over all triangles, circles, and squares found in the real-time image stream.
- **Post-Condition:** The application continues to apply the shape/color detection algorithm to the image stream until stopped by Mark.

#### II.B.2.8 Req. #8 - Pausing the Real-Time Camera Stream

- **Pre-Condition:** Mark has configured at least one camera and is now viewing the real-time stream.
- **User:** Mark clicks the "Pause" button in the camera control window. The real-time image is now paused, and will not refresh.
- **Post-Condition:** The application pauses and will not refresh the real-time image until Mark clicks the "Resume" button.

### II.B.3 Use-Case Diagram

The use-case diagram for the Silhouette Project is shown in Figure II.B.3.1. Actors are depicted using stick-figures, and various components of the system are shown in a bubble, connecting primary actors.

Figure II.B.3.1. Silhouette Project Use-Case Diagram.

