**Project 2: MCDA for Object Recognition**

**COSC 4207**

**Seminars in Computer Science**

Professor: Dr. Boguslaw Schreyer

By: Devin Rotondo

Nipissing University

April 3, 2014

Table of Content

[Introduction 3](#_Toc384228908)

[Steps for Object/Image Recognition 4](#_Toc384228909)

[MCDA Low-level Processing 5](#_Toc384228910)

[Colour Histogram 5](#_Toc384228911)

[Canny Edge Detector 5](#_Toc384228912)

[Pre-processing for the Canny Edge Detector 6](#_Toc384228913)

[Extracting the Length and Width 6](#_Toc384228914)

[Obtaining the Length/Width Ratio 7](#_Toc384228915)

[MCDA High-level Processing 7](#_Toc384228916)

[Normalizing the Data 7](#_Toc384228917)

[Comparing the Objects 8](#_Toc384228918)

[Weight Calculation 8](#_Toc384228919)

[The Ideal Point Method 8](#_Toc384228920)

[Ideal Point Method Image/Object Recognition Algorithm 9](#_Toc384228921)

[Program Features 9](#_Toc384228922)

[Establishing an Item Database 9](#_Toc384228923)

[Maintaining the Standardization 10](#_Toc384228924)

[Learning the Data 10](#_Toc384228925)

[Comparison Log 12](#_Toc384228926)

[Canny Log 13](#_Toc384228927)

[Interfacing with the Camera 13](#_Toc384228928)

[Web Images 13](#_Toc384228929)

[Program Classes and Key Functions 14](#_Toc384228930)

[Program Execution 15](#_Toc384228931)

[User-driven Execution 15](#_Toc384228932)

[Program Execution Chart 16](#_Toc384228933)

[Connect to Camera 16](#_Toc384228934)

[Reset 16](#_Toc384228935)

[Exit 17](#_Toc384228936)

[Run 17](#_Toc384228937)

[Results 19](#_Toc384228938)

[Conclusion 19](#_Toc384228939)

[Acknowledgments 20](#_Toc384228940)

[References 20](#_Toc384228941)

[Appendix 21](#_Toc384228942)

[Hardware Setup Guide 21](#_Toc384228943)

[AXIS Communication - AXIS Device Setup Guide using the AXIS IP Utility 21](#_Toc384228944)

[Connecting to the Nipissing University Wireless Network through the AXIS Camera 24](#_Toc384228945)

[Software Setup Guide 25](#_Toc384228946)

[Owoce (Fruits) Project Setup 25](#_Toc384228947)

# Introduction

Multicriteria decision analysis (MCDA) or multicriteria decision making (MCDM) is a decision making process in which a set of alternatives are evaluated based on conflicting and varying criteria. An Ideal Point (IP) method is a multiattribute decision rule of MCDA. A modified IP method has been used for object recognition of various images. An interactive MCDA-IP method was implemented in Java with a C# program for the online recognition of objects. Learning abilities have been incorporated through weights by employing a delta rule similar to that employed in some artificial neural networks. The learning is a new feature that is not usually employed with MCDA but does certainly help in a guided system. A wide group of fruits and vegetables are being recognized through the online and incorporated pictures from the camera. The ambiguities have been reduced due to the learning abilities of the system and user guidance for the initial setup or object loading.

# Steps for Object/Image Recognition

There are two categories of processing employed in object / image recognition. These are categorized as *low-level* processing and *high level* processing.

Low-level Processing

In the traditional object recognition, the *low-level processing* encompasses the following:

* Image capture - where the RGB image of the object is captured
* Image-preprocessing - where some undesired features can be removed from the image such as noise
* Image segmentation - where the image may be strategically separated from its surroundings
* Shape representation - where the shape of the image is obtained through an algorithmic method
* Description and physical feature extraction - where the inherent features of the image are obtain

High-level Processing

The high-level processing encompasses the following:

* A heuristic concept - where the result may be known and the object is evaluated based on this
* A priori knowledge - where prior knowledge from the problem is known and used
* Syntactic approaches - where conditions are used for comparison
* Semantic approaches - where features are used for classification

In MCDA object recognition, the processing varies from standard object recognition and can be seen below.

Low-level Processing

* Calculating a 64-dimensional image histogram - this computes a colour histogram
* Extracting features like length, width, and length width ratio
* Extracting physical features of the objects

High-level Processing

* Calculating performance or object features as to obtain a feature vector - this step uses the ideal point method decision rule

# MCDA Low-level Processing

## Colour Histogram

The colour histogram is calculated using a binning method. Each image will have at least Length \* Width pixels present, however, the binning method requires separating the intensities between the red, green and blue (RGB) levels. Each colour is separated as follows: intensities 0 to 63 make bin 1, intensities 64 to 127 are in bin 2, intensities 128 to 191 are in bin 3 and finally intensities 129 to 255 compose bin 4. Thus, each of the RGB values may be placed in one of four possible bins which gives a colour histogram composed of 43 or a 64 dimensional feature vector, Hik, where i is the current object to which the histogram is computed and k is the corresponding feature vector of the histogram k = 0, 1, ..., 63.

The following table illustrates the calculation of the feature vector and the corresponding histogram bin.

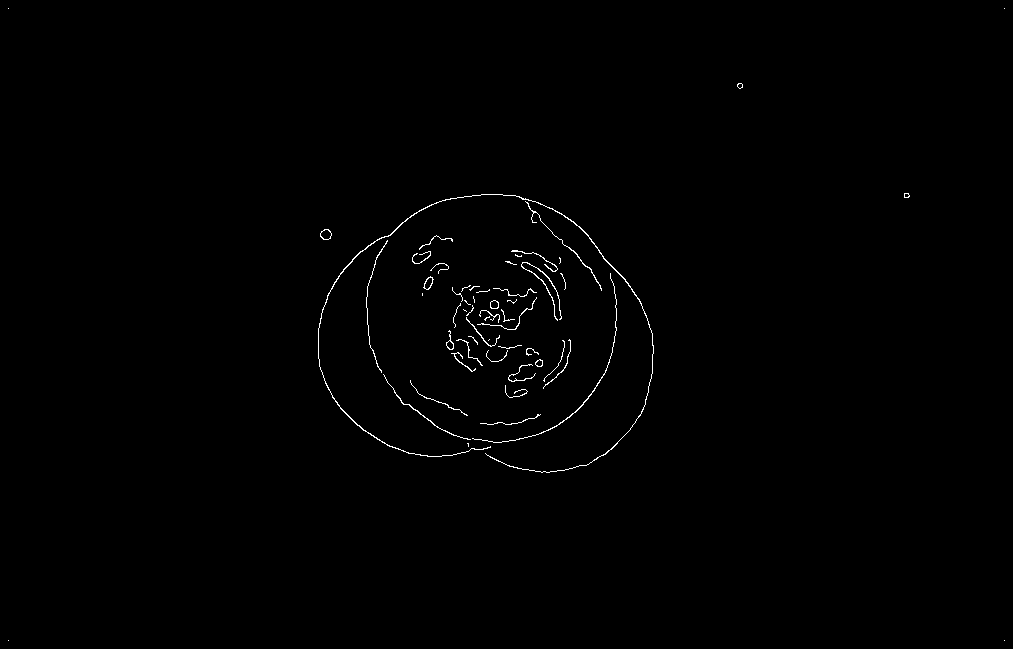
|  |  |  |  |
| --- | --- | --- | --- |
| Feature Vector | Bin # Red | Bin # Green | Bin # Blue |
| H0 | 0 | 0 | 0 |
| H1 | 0 | 0 | 1 |
| H2 | 0 | 0 | 2 |
| H3 | 0 | 0 | 3 |
| H4 | 0 | 1 | 0 |
| ... | ... | ... | ... |
| H63 | 3 | 3 | 3 |

## Canny Edge Detector

The main purpose of the canny edge detector is to determine the edges of the image so that the length and the width of the object can be obtained. However, a good canny edge detector will extract as many edges as possible. However, this is circumvented with two additional canny edge detector processing steps.

Before the canny edge detector is used, the current image is first transformed to calculate the perceived luminance. The technique employed, calculates the luminance based on the consideration of the biological perception of the eye sensitivity. In this case, the lowest sensitivity is thus to blue. The canny edge detector employed has two additional phases, the first is the non-maximal suppression stage. However, the canny edge detector first begins by generating a Gaussian kernel mask based on the inputted parameters for the canny edge detector and performs convolutions of this mask. Note that a blurring is also applied to ensure that neighbouring values are also used during the convolution. The non-maximal suppression is used to remove all values except for those that may be a local maximum. The local maximum value may or may not be an edge, so it is classified as an edge candidate. Once this has been done, the comparison is broken into several cases depending on the gradient direction. However, the specific algorithm used for this step instead employs the interpolation technique in which the partial derivatives are obtained and a comparison is done between them and the edge point to determine whether or not if it is an edge. Note that this also allows for simplifications in determining whether or not if the pixel is an edge since rotation and reflection may be employed reducing the complexity of the problem to several simple cases. Finally, the second phase of the canny edge detector is employed. This step is the hysteresis and thresholding. This step uses some previous knowledge, to be specific, it uses threshold levels in determining whether a value is an edge or if it is not. If the value is less than the specified minimal value, it is classified as not being an edge and if it is larger, it is classified as being an edge. If the value is between the minimum and maximum bounds, further comparisons are applied to determine whether or not these are edges or not. Overall, the canny edge detector performs well, but runs into issues when there is not high contrast between the object and background of the image. Furthermore, pre-processing is applied to remove the shadow of the objects as these are detected by the canny edge detector.

Below is the result of an apple put through the canny edge detector.



### Pre-processing for the Canny Edge Detector

Due to the issues with the shadows present during the canny edge detector processing, a pre-processing step has been applied. This pre-processing step removes light gray levels from the image below a certain specified threshold. The new result after the pre-processing is below.

****

### Extracting the Length and Width

Once the results from the canny edge detector are obtained, the length and the width of the object can be obtained. Note that since the length and width are arbitrary depending on the orientation of the object when the image is taken, the width is taken as the smallest dimension and the length is taken as the largest dimension. The dimensions are computed by taking the difference between the left edge and right edge and the difference between the top and bottom edge of the resulting canny edge detector image.

### Obtaining the Length/Width Ratio

The length/width ratio is simply taken as the ratio between the length and width as follow:

Length/Width Ratio = Length/Width.

# MCDA High-level Processing

## Normalizing the Data

The criteria used for the comparison: the length, the width, the length/width ratio, the weight and the histogram are not all measured in the same units. Length and width are based on pixel values, the weight is measured in grams and the histogram relies the 43 or 64 dimensional colour distribution value. Thus, to be compared, it is best to scale the criterion values to understandable and easy-to manipulate values by normalize them. The equation used for the normalization is as follows:



Cji is the ith criterion of the jth object. This criterion is normalized to give the following Cji ϵ [0,1] for each criterion. Note however that the largest and smallest value of the criteria are solely based on the items currently in the database. Unfortunately, this has led to some interesting challenges in determining these values. The technique used was to mark items when they were created. This was done by setting a changed flag if a new smallest or new largest value was obtained. Upon adding the item to the database, the change flag is used to update the value of all of the current items in the database. Note however, that since the maximal and minimal values will always be changing when the database is created, it was decided to store the original value for each criteria and to use global criteria values to update the criteria as needed. Another issue that also arose in this process was that since the minimum and maximum value of the criterion were specified as static (shared between all instances of the class), this value was not stored (serialized) when the database was stored to file. Thus, this feature has been implemented manually by storing these values when the program is closed properly (with the X) and by loading these values when the program is started. Note that when the program is reset, this criterion data is also reset.

## Comparing the Objects

It should be noted that the main goal of this project is to better compare objects. Thus, the actual criteria used for the comparison are the differences between the criteria for the current object and the jth object in the database. This difference is simply calculated using the Euclidean distance between criteria, note that the histogram is sent as a whole when compared. That is, instead of performing the Euclidean distance individually between its 64 criteria, it is performed as a whole resulting in simply one value.

## Weight Calculation

Once all criteria are normalized, the weights can be applied. Due to the nature of the current project, during my experimentation, it was found that the length and width should be regarded as less of a important since they are dependent of the objects placement to and from the camera and may not directly reflect an increase in mass of the object (bigger apple or smaller apple). The length and width were there for each weighed as 5% with the other 90% split evenly between the three remaining criteria - length/width ration, weight and colour histogram. Note however that further experimentation is needed in deciding the optimal balance between weights and criteria. Also, in the future, when new criteria are added, this will have to be looked at. However, for now, the results are promising with the current criteria and weight balance.

## The Ideal Point Method

The ideal point (IP) method is one of the many decision rules that are available for multicriteria decision analysis. The method categorizes objects based on their separation from a selected ideal point. The positive ideal point is usually ideal depending on whether or not the problem is to be minimized or maximized and on how the data are to be ordered - whether in ascending or descending order. The ideal point is indicative of a preferred alternative decision outcome. It consists of the weighed, *w* standardization value for each criterion, *C* of all alternate objects, *i*. In this case, the criterion refer to the length, width, length width ratio, weight and the 64-dimensional image histogram. The closest object to the ideal point is determined to be the best alternative and the separation from the ideal point is based on the results of a distance metric, *p* which in this case is 2 for Euclidean. The following is the formula for the separation from the positive ideal point to each alternative:



and the next formula is for the separation from the negative ideal point:



Note that in the first equation, C+j represents the desired value for the jth criterion or the ideal value and C-j represents the least desired value for this criterion.

### Ideal Point Method Image/Object Recognition Algorithm

The algorithm of the ideal point method employed specifically for the object recognition is as follows and takes into account that the database of objects has already been established and standardized.

Step 1 – Define the weights assigned to each criterion *j* as follows 0 ≤ *w*j ≤ 1 and 

Step 2 – A feature vector is the constructed and standardized by multiplying each criterion by its

weight for each object

Step 3 – The maximum value *C*+j of each weighed standardized criterion is taken as the ideal point for that criterion

Step 4 – The minimum value *C*-j of each weighed standardized criterion is taken as the negative ideal point for that criterion

Step 5 – The separation from both the positive and negative ideal are calculated. A similar formula used to compute the ideal point is used for the negative ideal point.

Step 6 – The distance to the ideal point is then computed as follows 

Step 7 – The results are then reordered according to their rank with respect to desired alternative

# Program Features

## Establishing an Item Database

The way that the program is designed is that when an image is submitted for recognition, a new item is created. This item will contain all of the criterion values for itself including its 64D colour histogram. There is an ongoing list of database items name Item.**extension** if an item is to be added to the database it's added to the list of Item.**extension**. When the program is exited normally, with the **x** icon or through the exit function, the **extensions** are saved and since the items are all serialized, this is done quite easily. Upon starting the program, the current database is immediately fetched and displayed. Note however that after fetched, and displayed, the sum values are meaningless since no comparison is taking place. However when loaded, this allows to view the standardized object values between the objects currently in the database. These vary from 0 to the criteria weight or may be slightly higher depending if learning has been applied.

## Maintaining the Standardization

The standardization depends on a static variable shared between all items of the class. This value is reset at every program launch. This originally caused a problem so when the program is closed and opened these values are now stored and loaded. These value can also be reset when the database is reset. The standardization values are only updated if a new item is added to the database with either a smaller or larger value for any of the criterion already present. An original value for each criteria is stored when the item is created to avoid issues that would arise upon changing the minimum or maximum values.

## Learning the Data

Since the item, length, width, length/width ratio and weight are independent when the distances are compared, they can be dynamically modified to better resemble the qualities of the desired item to learn. An example of this is when an apple is submitted and does not match an apple already in the database, the properties of the desired apple can be used to modify the properties of the database apple through learning. The values are modified/learned using a variation of the delta rule employed for the Adaline artificial neural network. This has been employed to make the values of the current item only as small as needed to make the apple recognized as the best alternative the next time it is submitted. This learning aspect allows for the system to demonstrate more intelligent behaviour and will allow for objects to be better compared assuming that they have a similar colour histogram. The issue in learning for the histogram is that the following formula is used to compute the differences between histograms.



In this case, j represents the index for the current object when compared to the object that has been submitted. The histogram is the result of the Euclidean distance between the 64D histograms. Note that this can be viewed as follows









Note that although some of these values can be stored, the values from the 64D histogram are still required and unless each of the 64 features are used, learning is not possible. Thus, learning has not been applied to the histogram.

Suppose that you currently have the following items currently loaded in the database.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Item** | **Length** | **Width** | **Length/Width Ratio** | **Weight(g)** | **Histogram** | **Sum** |
| Peach | 144 | 104 | 1.38 | 200 | 64D |  |
| Kiwi | 100 | 20 | 5 | 100 | 64D |  |
| Grapefruit | 200 | 180 | 1.11 | 315 | 64D |  |

Now suppose that the following item is submitted for recognition.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Item** | **Length** | **Width** | **Length/Width Ratio** | **Weight(g)** | **Histogram** | **Sum** |
| Apple | 240 | 200 | 1.2 | 270 | 64D |  |

Now, since the grapefruit has the closest resemblance to the apple, it will be taken as the best alternative for the submission. However, suppose that the peach is actually the item that as a user you believe should be the best alternative.

How would the learning have to be modified? As previously stated, a delta rule is applied for learning. Note that the delta rule is the same rule applied in the Adaline artificial neural network. Note when the learning occurs, the best sum (smallest sum) and the properties of the current object are sent to the desired match. The learning occurs until all criteria attain the desired value, until a smaller sum is obtained or until the maximum number of iterations is complete. As an example, let's look at learning applied between the apple and the peach for the first criteria (length).

LengthAPPLE- LengthPeach\*Weight1 =

240 - 144\*1 = 96 \* 0.1 = 9.6

Thus, after the new weight is applied, the peaches criteria value should increase by 9.6.

Thus, the weight is increased by = 9.6/144 =>

The weight is incremented by 0.0666

Thus, the difference is 240 - 144\*1.0666 =

Decrement the weight by 0.0433

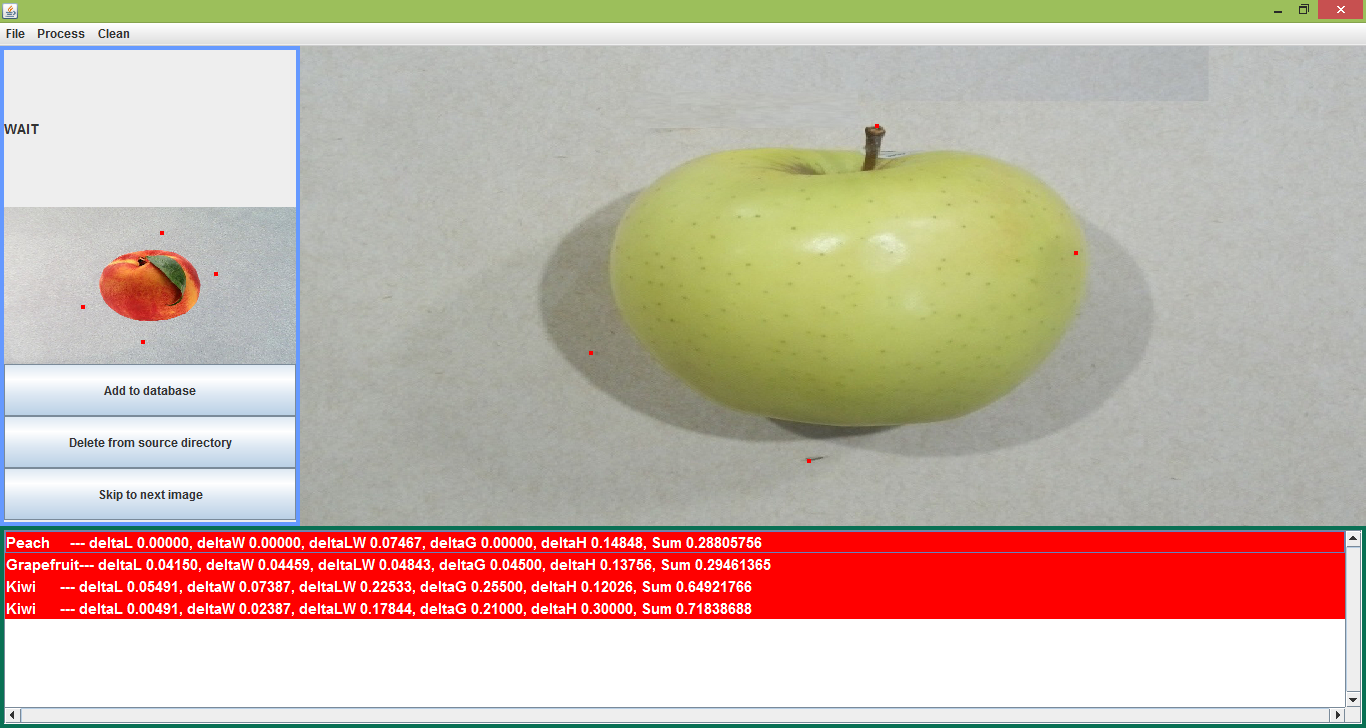
144 -240\*(1-0.0433) = 86.4. Thus, the system would have learned as the values are now slightly closer. This is repeated until the stopping condition is met.

Before Learning

****

After Learning - Fuzzy learning

This learning has been done with smaller weights and allows for more variation between the results, but as mentioned before, the learning is highly dependent of the histogram results.



## Comparison Log

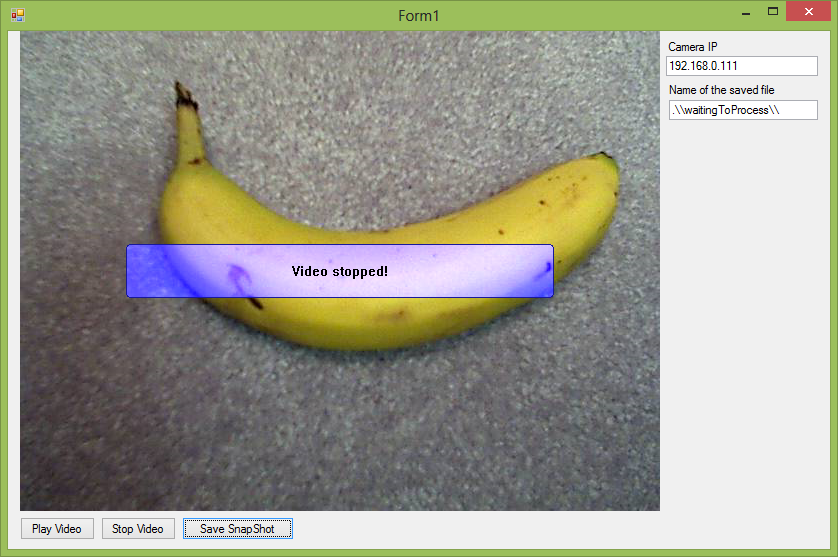
Although this feature is not an essential one, it is usually very important to log some information during the execution of a program. In common programs, this may vary from creating a debug log to user preferences. For the current program, the log stores the top three matches for each comparison. It also stores the data and time at which the comparison took place. This would allow for a user to keep a record of the previous comparison results as they disappear when the next object is input into the system. This can also be used for comparing learning outcomes based on previous trials, that is determining whether or not if the system is performing better or worse for previous submitted items after going through the learning. In the future, this may also be used by a user to obtain a profile of the systems results by automating a program to extract the output of the system log. The log is never deleted unless if the user manually deletes it, and the data is comparison information is constantly added to the start of the log.

## Canny Log

The Canny log is an output of the canny edge detector edges after the edges are returned from the canny edge detector. From the log, the ideal parameters were chosen for the current canny edge detector employed in the program. Although the results are not always as expected, they usually still have the general outline of the object.

## Interfacing with the Camera

The current program does not run in real-time, however, it may still use input from an external device. In this case, the program interfaces with the AXIS camera (See the appendix for setting up the camera). This is not a direct relation but the camera can be launched from the program interface by selecting Process → Connect to Camera. This will bring up the camera.exe program which is as follows:



To correctly interface with the camera, the IP address must be inputted. When inputted, the user will be prompted for a password which when input correctly will display the video from the camera. Some common features are available including Play Video, Stop Video and Save Snapshot. The Snapshot is used to interface with the Java Owoce program. Note that the name of the saved file should be kept with the current directory. This is because the Owoce program has a listener attached to the waitingToProcess directory and pulls objects from this directory when the processing is done.

## Web Images

Images of objects from the web may also be used with the application so long as they are saved in the waitingToProcess folder. Note that these images must also have the same dimensions as the objects obtained from the camera or else, the recognition will not go as expected.

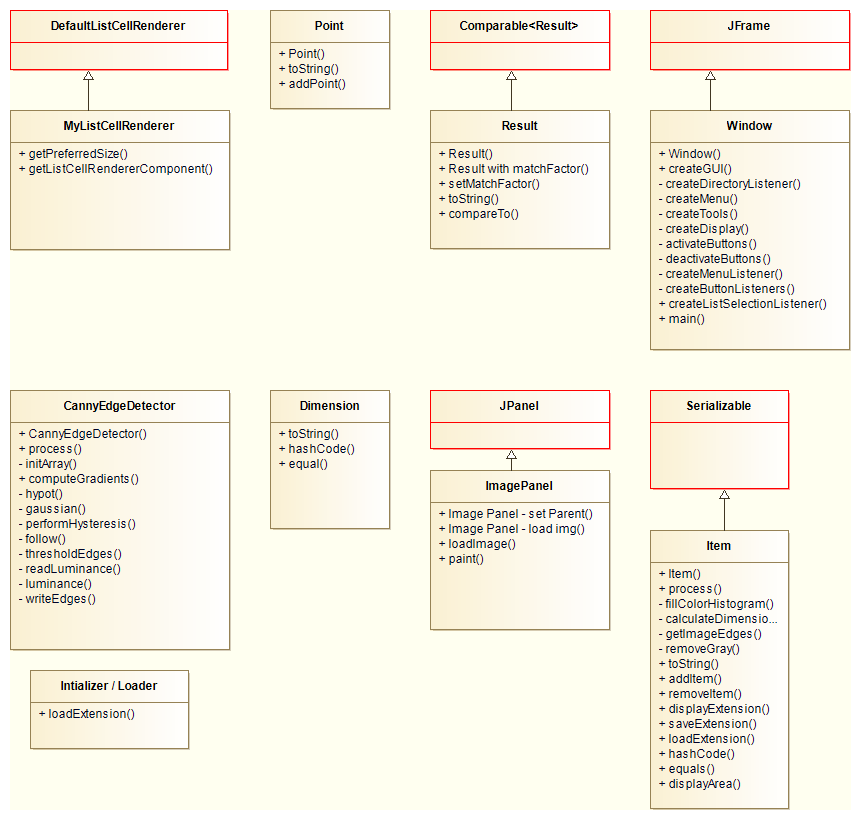
# Program Classes and Key Functions

Currently, there are 8 of these classes present. These are the following:

|  |  |  |
| --- | --- | --- |
| **1** - CannyEdgeDetector.java | --------------------------------- | Public class CannyEdgeDetector |
| **2** - Dimension.java | --------------------------------- | Public class Dimension |
| **3** - ImagePanel.java | --------------------------------- | Public class ImagePanel extends JPanel |
| **4** - Item.java | --------------------------------- | Public class Item implements Serializable |
| **5** - MyListCellRenderer.java | --------------------------------- | Public class MyListCellRenderer extends DefaultListCellRenderer |
| **6** - Point.java | --------------------------------- | Public class Point |
| **7** - Result.java | --------------------------------- | Public class Result implements Comparable<Result> |
| **8** - Window.java | --------------------------------- | Public class Window extends JFrame |

Out of all of these classes and files, Window.java contains the **main** function. Please see below for certain key functions present in each class.

+



Although these classes are all present, they are not executed at the same time and are not run in this order and are in fact user driven.

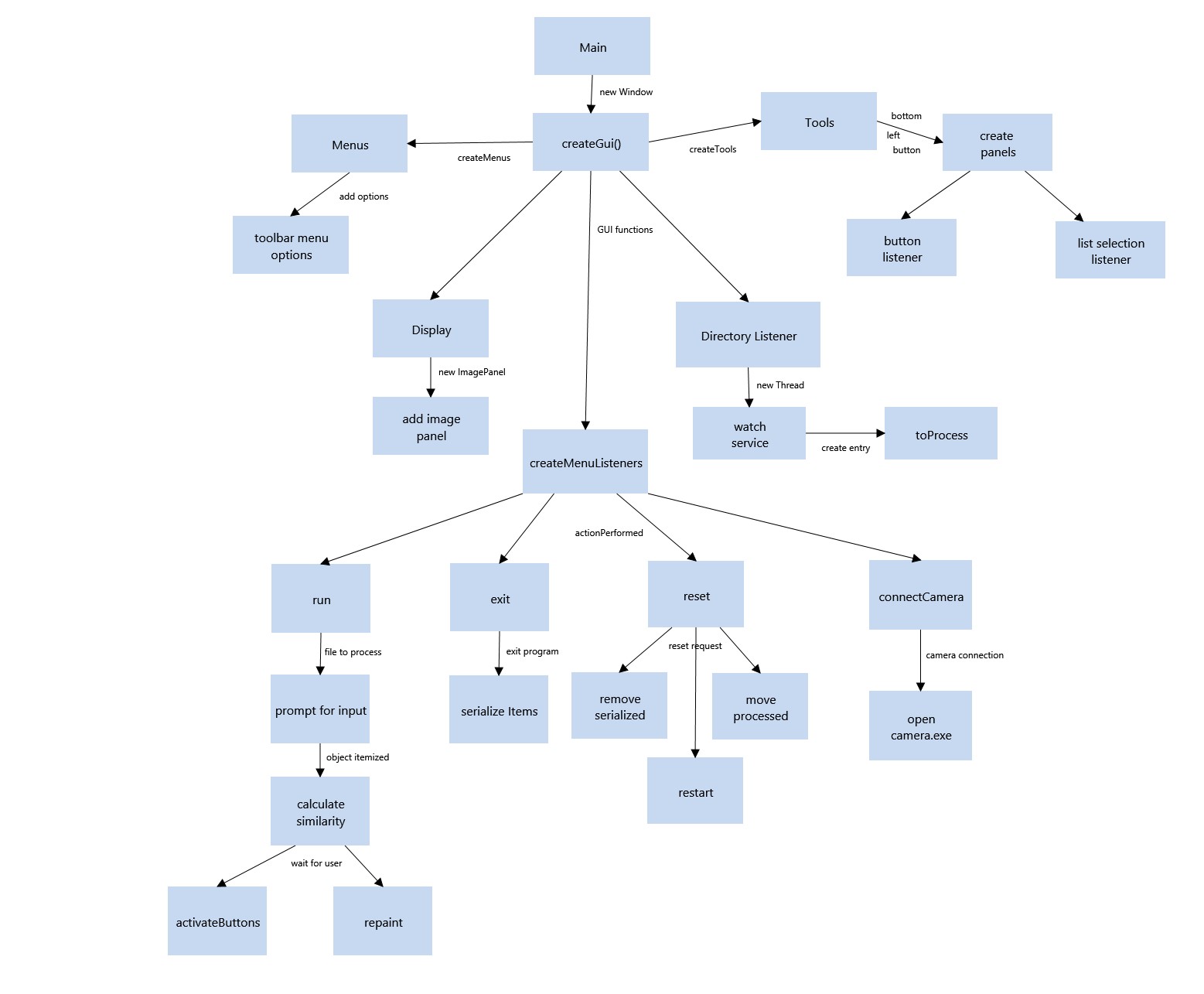
# Program Execution

## User-driven Execution

The Owoce Ideal Point program is completely driven by the user. Quite a few features are available to the user throughout the program execution.

### Program Execution Chart

The following diagram illustrates the sequence of events executed when the program is started and also has some of the user functions included, although abstracted under createMenuListeners.



### Connect to Camera

This option has already been mentioned, see Interfacing with the Camera.

### Reset

This option resets the database and standardization values.

### Exit

This option is used to exit the program normally.

### Run

Upon pressing the **Run button** or employing the hotkey **Ctrl + R** the following code is executed proceeding the run.actionListener event

**if**(!toProcess.isEmpty()) {

currentAnalyzedFile = toProcess.pop();

It is important to know that here, toProcess refers a stack of files corresponding to the image files present in the waitingToProcess folder

Following this, the image is displayed in the larger of the two image panels

imagePanel.loadImage(currentAnalyzedFile);

Note that the function that is now called is taken from the ImagePanel class and uses the built in read function for buffered images and also obtains the width and height of the image.

The user is then prompted for the name and weight of the object.

Following this, a new serialized Item is created.

currentAnalyzedItem = **new** Item(currentAnalyzedFile, itemName, itemWeight);

Note that this function calls the constructor from the Item class. A special feature of this construction is that the constructor of the superclass (Serializable) is also called.

-------------------------------------------------

Steps performed by the Item class

-------------------------------------------------

During the Item constructor some private variable details are set and the process() function is called.

In this function, the BufferedImage is read a new -> this can therefore be optimized. The "gray" of the image is removed when all RGB values are > 100. That is R > 100, G > 100 and B > 100. These pixels are then coloured black for them to be processed easier by the canny edge detector.

removeGray(image);

BufferedImage edges = getImageEdges(image);

The getImageEdges function employs the CannyEdgeDector class, calls its constructor and sets the source image and uses its process function

---------------------------------------------------------

Steps Performed by the Canny Edge Detector

---------------------------------------------------------

The process() function obtains the width and height of the image, populates private values of the canny edge detector. The luminance of the image is also obtained by reading all of the RGB values and applying the perceived luminance function to the value triplets.

Luminance (perceived option 1): (0.299\*R + 0.587\*G + 0.114\*B)

An important part of the canny edge detector is performed during the computeGradients function. During this stage of the canny edge detector processing, a non-maximal suppression is performed. That is, only preserve local maxima values and remove false positives or troublesome values.

In the next step, threshold edges, all edges are specified and in the write edges, writes out the resulting edges.

----------------------------------------------------

end of the Canny Edge Detector Processing

----------------------------------------------------

As soon as the Canny Edge Detector has completed the width, length and length to width ratio are obtained.

The colour histogram is also calculated using 64 bins.

The item is then declared as processed.

-------------------------------------------------

end of the Item class processing

-------------------------------------------------

Following this, the calculate similarity function is calculated (this is done using Euclidean Distance specified in Art 2). This step compares the new item with all other items in the database.

The dimensional data are compared (length, width, length width ratio, weight and colour histograms) with every other item that is presently stored in the database using the ideal point method and a result is calculated for each item as a weighted sum.

The results are then sorted and added to the list model and are therefore displayed since the model is associated with the JScrollPane on the bottom of the screen.

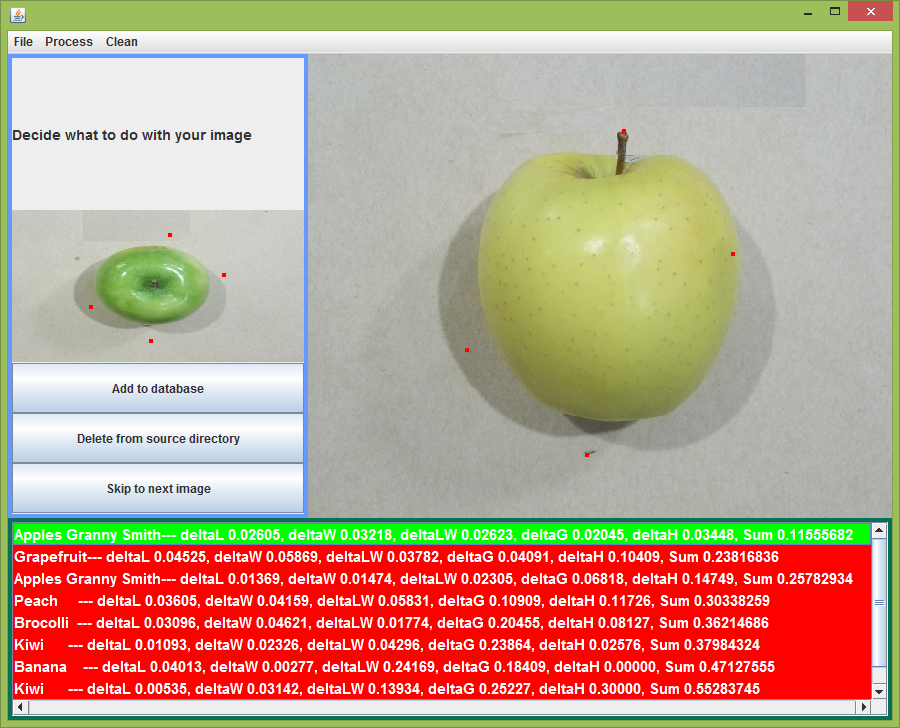
The buttons are then activated for the user to select what to do with the image.

Upon pressing the add button, the item is copied over to the processed folder and the run function is once again called. Upon right-clicking the mouse on a specified item, learning is used.

When the user closes thee application, the entire database of items currently loaded is saved.

# Results

Upon running the program and comparing the items a resulting set of alternatives are displayed. These may appear as follows:

****

In this case, a yellow apple has been successfully recognized as a granny smith apple. This demonstrated some of the learning capabilities of the system. The green colour represents that the current alternative is within a threshold of ideal matches. This value does not change over time, but may be improved through the learning capabilities of the system.

# Conclusion

The MCDA approach using the ideal point method provides promising results. The application is recognizing a wide variety of objects and is able to identify the same object with some varying criteria as identical. Future research may include extracting objects within more complex images based on known properties of the image (E.g. Placement of the object, contrast) using . Additional criteria will also improve the performance of the application and the effects of varying the weight of the criterion may also prove to enhance the recognition.

# Acknowledgments

The author would like to thank Wojtek for previous project contributions and Dr. Schreyer for guidance and support.

# References

Malczewski, J., GIS and multicriteria decision analysis, JOHN WILEY& SONS, INC, 1999

Schreyer, B., "Site selection for light manufacturing development with multicriteria decision analysis: A case study of municipalities in Ontario, Canada"

Schreyer, B., "OWA-inspired object recognition", Artykul 1 MCDA inspired object recognition

Schreyer, B., "OWA-inspired object recognition", Artykul 2 MCDA inspired object recognition

Schreyer, B., " Ideal Point method for object recognition", Artykul 3 Ideal Point method for object recognition v2

# Appendix

## Hardware Setup Guide

### AXIS Communication - AXIS Device Setup Guide using the AXIS IP Utility

1) Make sure that both your computer and the camera are connected to the same network and that you have the AXIS IP Utility (Single camera installation) or the AXIS Camera management (Multi-camera installation) software installed. The software used for this installation is the AXIS IP Utility. Also, if this is your first time using the camera on the **current network**, the setup must be done using a **broadband connection**.

**Please note :** If you have not yet setup the camera or that the camera details have been inherited from another user and that the username and password to connect to the camera are not known, a system reset is needed.

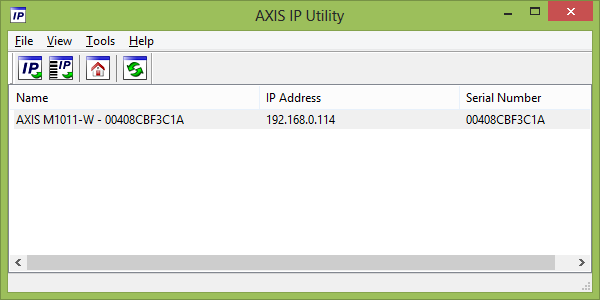
**Resetting to the Factory Default Settings** :

* Disconnect power from the camera.
* Press and hold the Control button (button on the back of the camera) and reconnect power.
* Keep the Control button pressed for about 30 seconds until the Status indicator flashes amber.
* Release the Control button. The process is complete after about 1 minute (when the Status indicator turns green). The camera has then been reset to factory default settings.

Now open the AXIS IP Utility Program and the device **should appear** in the window.

If the AXIS camera **does not appear** in the IP Utility, many problems may be present with network security. Thus, if on the school network, since the wireless and wired networks are separate make sure that you are on a wired computer with the camera plugged in using the broadband connection to the same network. If this is done, the use the IP Utility from the local computer. Run the IP Utility and wait to see if the camera shows up. If it does not appear after a while close the program and right-click the IP Utility program and select Troubleshoot Compatibility and test with the given selection and make sure that the program is given access to the wireless network if prompted. If again the camera does not appear, go to Control Panel → Network and Internet → View Network Computers and Devices and manually obtain the device IP and input it in the IP Utility. If still not found just paste the IP Address in the internet search bar and you should be on the cameras' homepage.

Once properly **setup**, the AXIS IP Utility will now be able to see your AXIS device. See the image below.



Now **double click** on the device.

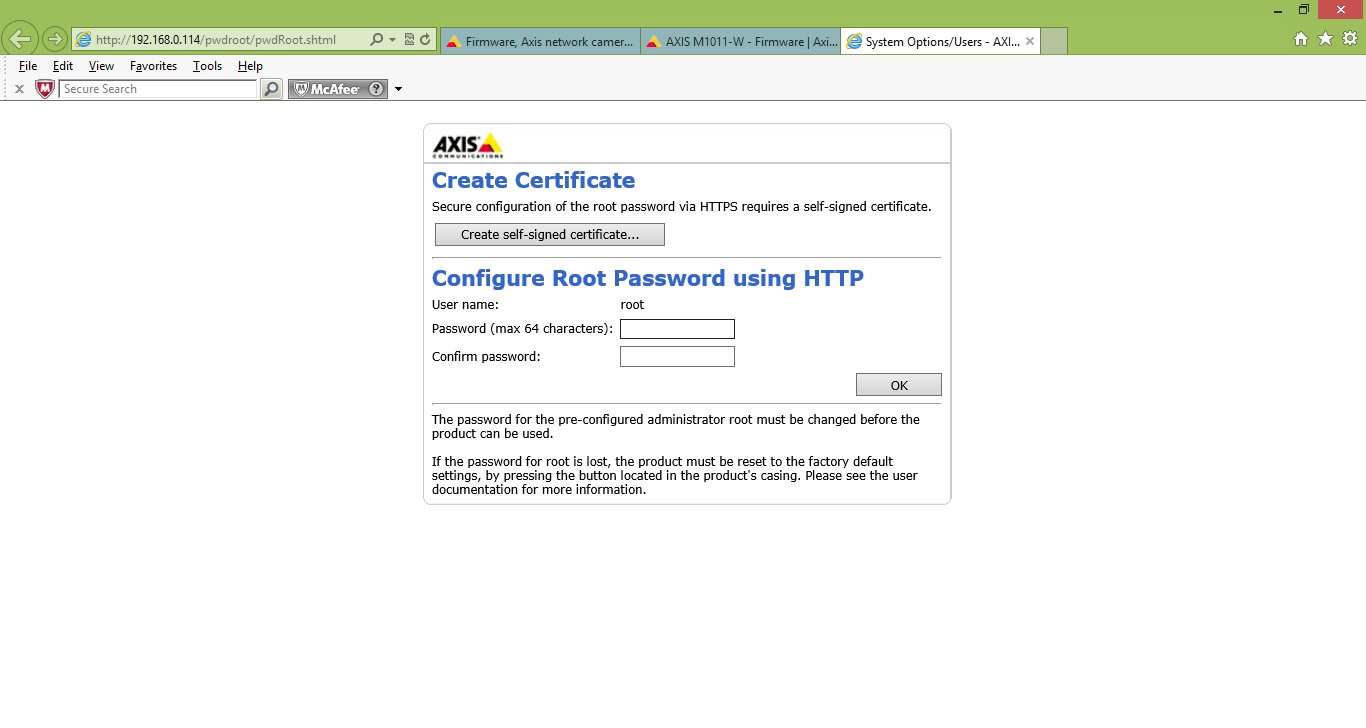
You will be redirected to the **cameras webpage** (these details are stored in the camera).

Here, if you just reset the system, you will have to create a **username** and **password** for the camera.

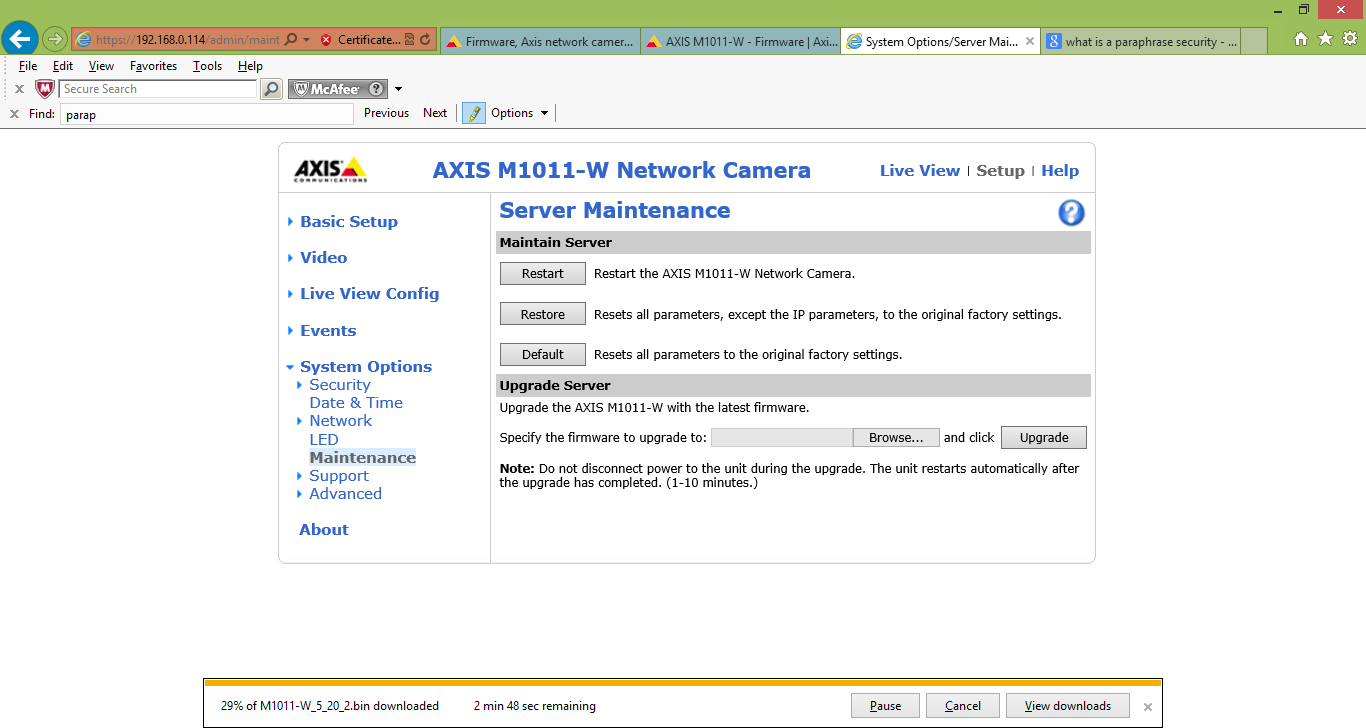
If these details have already been setup, input them and proceed to the next step.

Now that you are on the camera webpage, select **Network** and input the appropriate security type and password for your network (if any are present) to allow for a **wireless connection** to be established.

Now, to avoid eaves dropping, enable the HTTPS extension by creating a self-signed certificate. This enables the live feed being sent and received and all other communication by the camera to be encrypted. If you are not happy using HTTPS, you can also add another account to the device which will allow for a regular connection to be established.



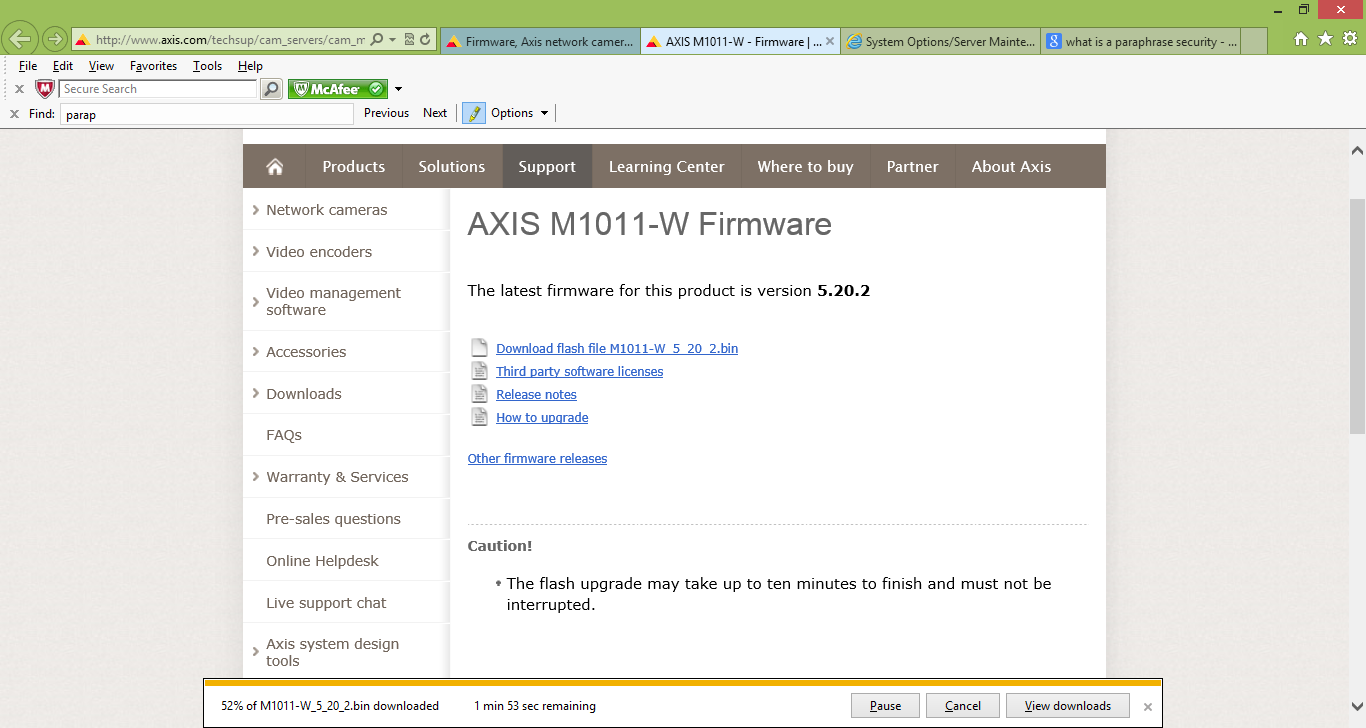
Since the camera has now been setup, a **firmware update** is now needed. This will ensure that the camera is running the latest software and that all compatibility issues are fixed.



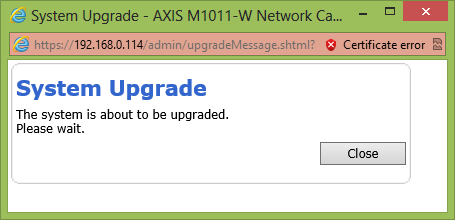
The latest firmware that was found was located on the AXIS website and appeared as follows

http://www.axis.com/techsup/firmware.php

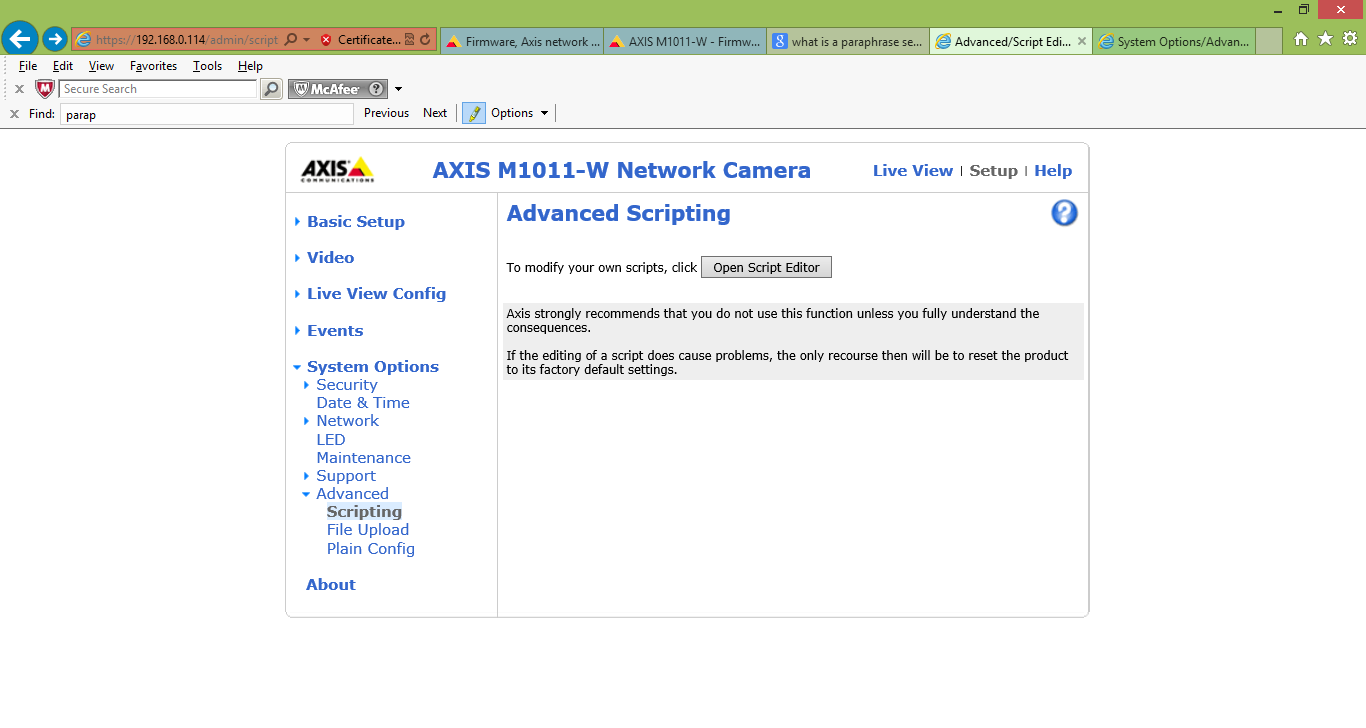
http://www.axis.com/techsup/cam\_servers/cam\_m1011w/firmware.php --- for the specific camera used in this example



When upgrading, the following will appear



An interesting note before ending the setup tutorial is that the AXIS camera appears to also be able to run live scripts. Please see below.



File and SDK setup

http://www.axis.com/techsup/cam\_servers/dev/index.htm

To use the camera wirelessly, ensure that the broadband connection is removed.

Congratulations, your AXIS device is now ready for use and has

been formatted with all of the latest firmware and security features

### Connecting to the Nipissing University Wireless Network through the AXIS Camera

Note that since the main goal is to be able to use the camera wirelessly, the camera should be connected to the correct network. This is accomplished by connecting the camera to the nuwlnet (the Nipissing University Wireless Network). It has been tested and the connection indeed still works correctly when not in the Sun lab. Specifically, the camera has been accessed from the share Nipissing and Canadore without any problems.

## Software Setup Guide

### Owoce (Fruits) Project Setup

**Requirements** - Assuming that the camera is fully functional

 1. Get and install the JDK proper for your OS version:  <http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html>

2. Get, unzip and start an Eclipse Standard in a current version:

<http://www.eclipse.org/downloads/>

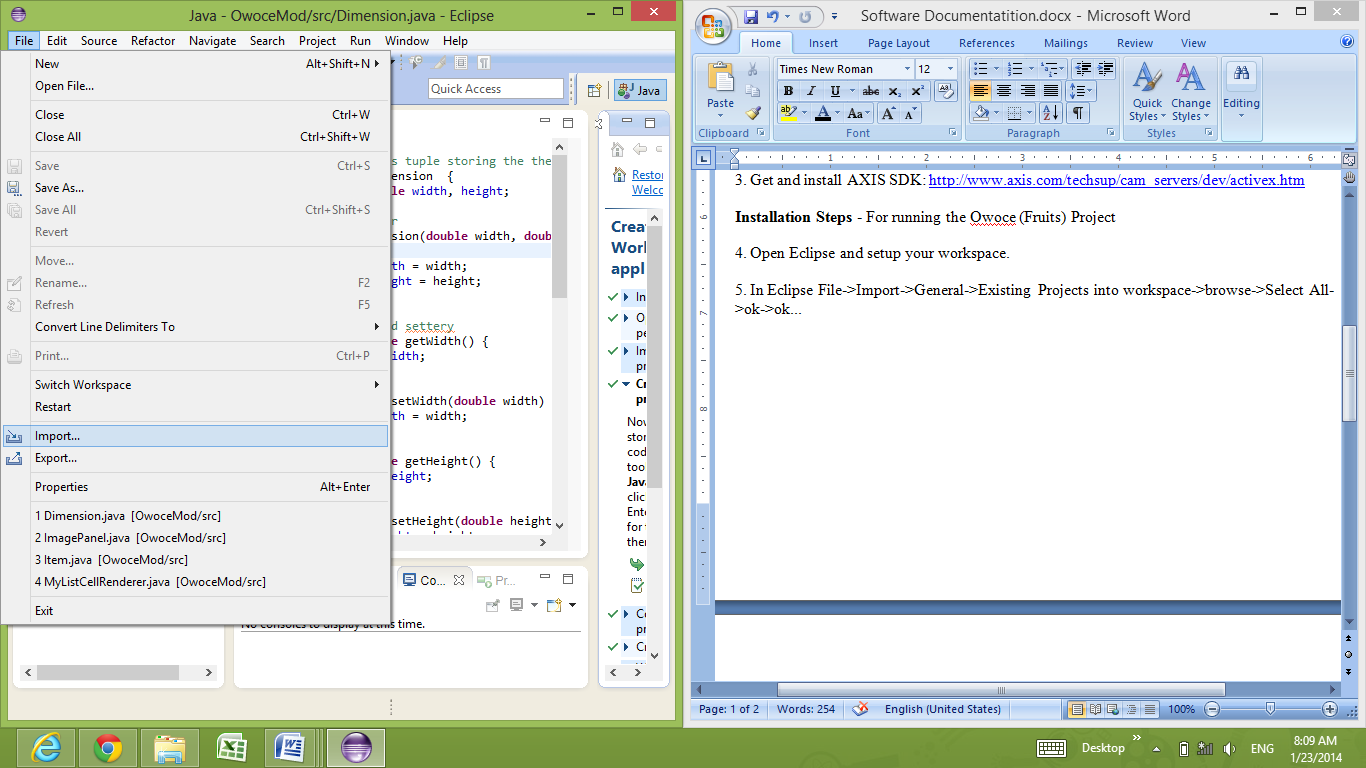
3. Get and install AXIS SDK: <http://www.axis.com/techsup/cam_servers/dev/activex.htm>

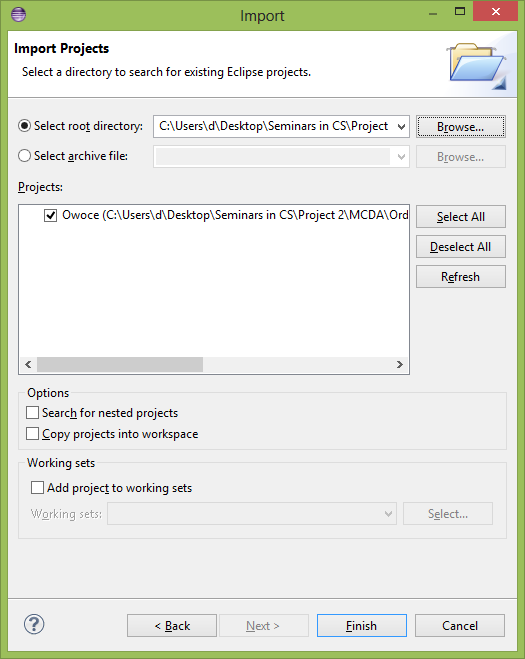
**Installation Steps** - For running the Owoce (Fruits) Project

4. Open Eclipse and setup your workspace.

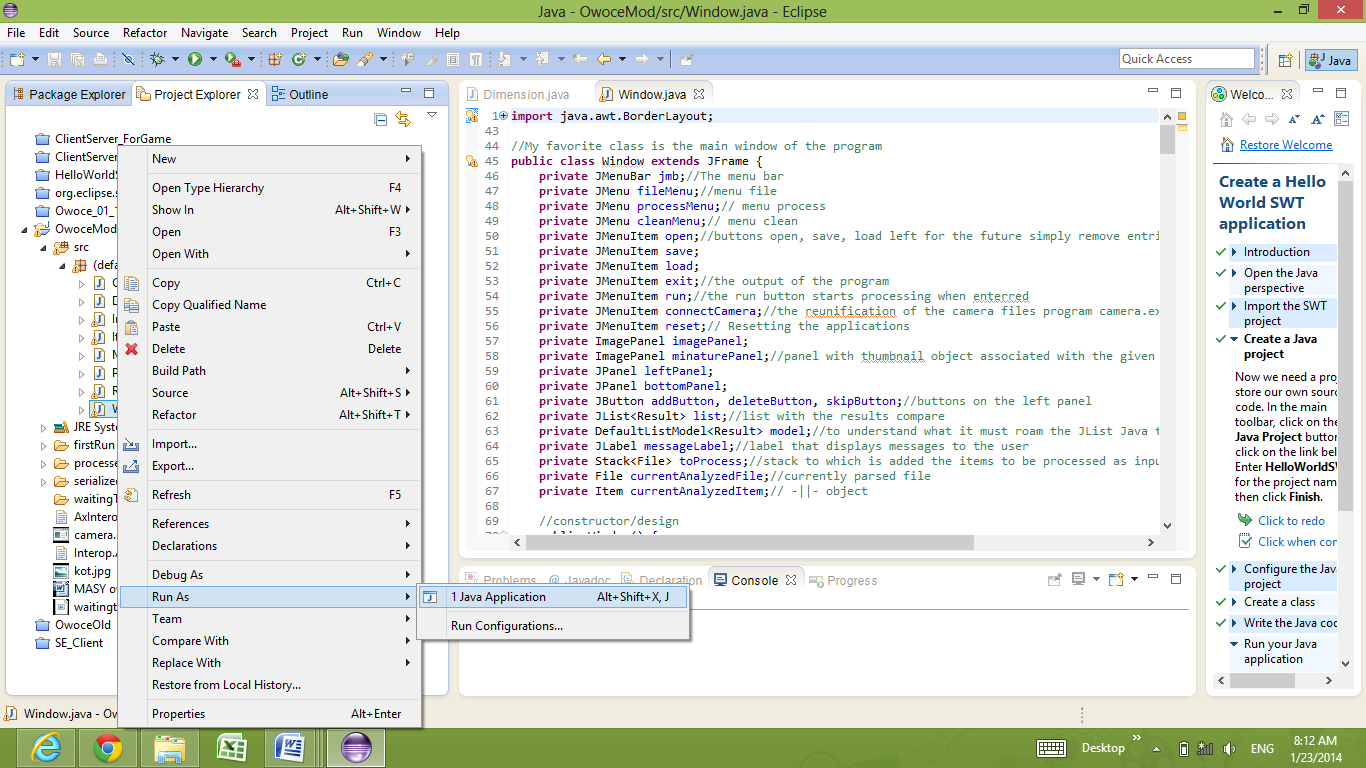
5. In Eclipse File->Import->General->Existing Projects into workspace->browse->Select All->ok->ok...

See below for the graphical example.

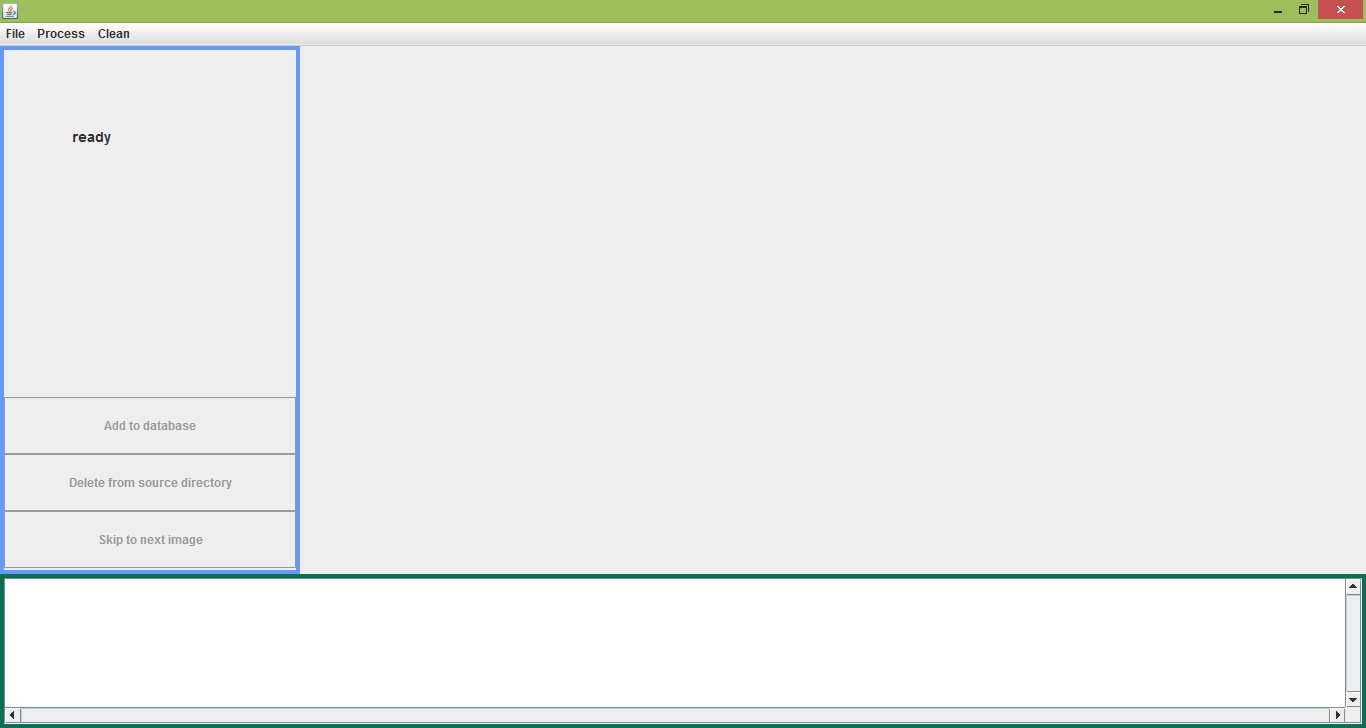


****

6. Open the class file Window.java in explorer and run the Owoce (Fruits) program.



You should now see the following:



7. If the program does not work, install the newest version of .NET environment

-----------------------------------------------------------------------

**Basic Details on the Program**

In the Owoce program folder, there are four (4) sub folders.

The folders ***waitingToProcess***, ***processed*** and ***serialized***are important during the programs execution.

--------------------------------------------------

**Folder Details**

The ***waitingToProcess*** folder : contains all files that are waiting to be processed. Whether for addition to the database or to compare items.

The ***processed*** folder : contains the files that have previously been processed

The ***serialized*** folder : contains the serialized program objects.

**Note** that the files read in by the Owoce program are of type JPEG and can be obtained from camera snapshots.

------------------------------------

Processing the objects: run->process

**Note** that processing includes adding the object to the database or comparing a new object with the already processed objects.

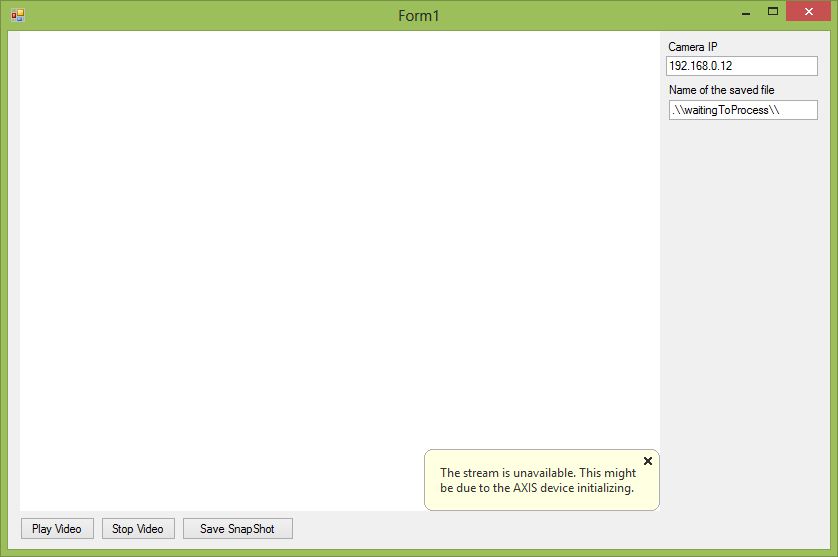
A key reset in a menu clean organizes and resets the application to an initial state.

--------

**Taking a Snapshot with the Camera**

To connect to a camera :

1) While the program (Window.java) is running press: process->connect to camera



The window above should have appeared when after this step has been executed.

2) You need to now give the correct IP address for the camera.

The IP is found by looking at the IP displayed in the AXIS IP Utility application.

3) When the IP is set, press play video, and you will eventually be asked for your password.

Once inputted correctly, the live video feed will be displayed.

---

Note that the following demo is a basis for the camera execution program.

AMC\_demo.zip camera control in C#.