Fruit Counting

Abbie Wade(u5182631) and Nicholas Mobbs (u5388261)

Last updated: 20:16, Sunday 7th June, 2015

1 Introduction and Related Work

In this project we were presented with multiple video clips of a conveyor belt carrying five types of fruit: Bananas, Oranges, Tomatoes, Pears and Green Apples. For each video we were required to detect all types of fruit in each frame of the video, track the fruit, correctly classify the fruit and count each piece of fruit exactly once. The purpose of this system is to develop a pipeline that could track and count objects given significant differences between each object which needs to be counted for statistical reasons.

Our implementation uses a combination of a soft minimum thresholding to remove noise, watershed filtering to separate and minor morphology to allow for easy identification of objects. After objects have been identified, through the use of contours and HSV values of the image, objects were classified according to predetermined values. Each classified object was counted in real time with edge detection.

While our implementation is a simplistic example, there is potential in real world applications as this project is considered a toy example of traffic monitoring systems. In the real world this could be used in a traffic system for occurrences of bridges or tunnels. Generally specialty roads such as bridges and tunnels have a load limit which needs to be adhered to to avoid the road breaking under the stress of the weight. There are many different types of vehicle classes that use the roads each day, all with different weights. A motorbike is lighter than a car or truck for instance. Our simplistic algorithm presented in this report could be modified to identify the different vehicles using the specialty road, track the vehicle and count the number of vehicles currently using the specialty road. It could then go the step further and calculate the average weight to assist in controlling traffic to avoid a stress breakage of the road.

1.1 Online Tutorials

The most valuable references to help start this assignment has been the documentation for openCV found at:

- OpenCV 2.4.11.0 documentation. 2015. OpenCV Tutorials ? OpenCV 2.4.11.0 documentation. [ONLINE] Available at: http://docs.opencv.org/doc/tutorials/tutorials.html. [Accessed 07 June 2015].
- OpenCV-Python Tutorials – OpenCV-Python Tutorials 1 documentation. 2015. OpenCV-Python Tutorials – OpenCV-Python Tutorials 1 documentation. [ONLINE] Available at: http://opencv-python-tutroals.readthedocs.org/en/latest/py_tutorials/py_tutorials.html. [Accessed 07 June 2015].

This set of documentation contains both all the inbuilt functions and tutorials for different image processing techniques which are current and up to date. The is a primary source which was used to preprocess the frames of the video before classification.

1.2 Research Papers

There are many papers that talk about the best way to classify fruit. Gill et. al. (2014) reviews about soft computing techniques to automatically classify fruit to provide better quality produce to the consumer. In this paper, soft computing refers to fuzzy logic through this use of models like neural networks or evolutionary algorithms to decide what fruit classification should be determined. It also talks about classification of different fruits, their accuracy, and what the best way to classify them is. Kodagali and Balaji (2012) discuss ways to talk about the variation in colour and texture of different fruits and the best way to deal with this. This paper also introduces the idea that fruit can have multiple classification combinations. Seng (2009) summarises research done by classifying fruit using the mean colour, shape and extent of the fruit in a more robust way. Using these methods, the system was approximately 90% accurate.

- Gill, J., Sandhu, P. S., and Singh, T. , 2014. A Review of Automatic Fruit Classification using Soft Computing Techniques. International Conference on Computer, Systems and Electronics Engineering (ICSCEE'2014) , 1, -.
- Kodagali, Jyoti A., and S. Balaji. "Computer vision and image analysis based techniques for automatic characterization of fruits?A review." International Journal of Computer Applications 50.6 (2012).
- Seng, W.C., 2009. A New Method for Fruits Recognition System. Electrical Engineering and Informatics, 2009. ICEEI '09. International Conference, 1, 130 - 134 .

2 Our Approach

As both our members come from computer science backgrounds and had no experience with MATLAB, we elected to use the more familiar design environment of python. This allowed the use of the heavily optimised libraries of opencv and numpy and the object orientated paradigm.

At a high level algorithm involves minor preprocessing to remove noise and segment objects, followed by some object detection followed by classification and counting of the objects found. The psuedocode for the algorithm is as follows:

```
while not video.end:
        for pixel in frame:
          if pixel.value < 100:
            pixel = 0
        frame.blur()
        for pixel in frame:
          if pixel.value != 0:
            pixel = 255
        frame.watershed()
        frame.dilate()
```

```
for cnt in frame.contours():
  if cnt.size > 20 and cnt.size < 800:
    fruit = cnt.classify()
    if fruit.y > line_start and fruit.y < line_end:
      if fruit not in current_fruit:
        count[fruit.type] += 1
```

For further details about each part of the algorithm, see the subheadings found below.

2.1 Preprocessing and Object Detection

Preprocessing is done in order to remove noise and to separate objects which are touching in the original video frame. We tried several approaches which included mostly morphologies and thresholding which was not highly successful in both removing noise and separating objects. The most effective method for preprocessing was found to be a simple soft-min threshold which floors values below the threshold, followed by applying a blur effect directly after to reduce the holes in the objects. This combined together preserved most of the information in the frame while removing both the noise and the conveyer belt.

For the problem of object segmentation we decided to use the watershed algorithm. Opencv uses a marker based watershed so we found the local maxima gradients using distanceTransform and then fed this into the watershed algorithm. After an additional threshold by a fudge factor (related to the mean distance from the local maxima) we were able to remove the bridges consitantly. However watershed often did not work well for non-spherical objects and ended up segmenting bananas into multiple parts. We overcame this issue by dilating the result to rejoin the objects.

Since we are now dealing with a binary image we're able to easily find the contiguous sets of pixels set to 255, we achieved this by using opencv's contour detection. Contours have some nice properties about them stored such as the size and extreme points. These features were used to classify the objects by looking at their mean BGR and HSV values and the extent of the contours. A simple example would be that a banana is not spherical so it would have a low extent.

2.2 Fruit Classification

For the classification of fruit we made a decision tree to classify the fruit by type. It used features of the contours introduced in the previous step, including information generated about the mean BGR and HSV values, size and extent of the object. Using this information we were able to determine the following about the fruit:

Banana	extent < 0.5
Tomato	mean red colour < 100 and mean blue colour > 230 and hue < 30 and saturation > 160 and size of object < 190 pixels
Pear/Apple	mean red colour > 30 and mean green colour > 220 and hue < 100 and saturation < 200 but > 135 and value > 208
Orange	mean red colour < 75 and mean blue > 210
Other	otherwise

The decision tree was determined in order shown in the table above.

2.3 Counting Fruit

In order to count fruit we checked if the centre of the object was in a rectangular area towards the bottom of the screen. If the objects centre was found to be located in this region, we incremented the counter for the fruit type the object was classified as. As soon as the object was counted, we added a tuple of the area and frame count remaining to a list. If the centre of a object was found in the rectangle again within the area (plus or minus a threshold) found before then we did not increment the counter.

3 Our Implementation and Experiment Results

Our implementation successfully counts bananas, oranges and tomatoes for the first three video clips. It is unable to deal with multiple fruit on top of each other. It is also unable to distinguish between green pears and green apples. As such, all green apples are counted as though it is a pear. Additionally our implementation beeps every time an 'other' object is detected.

3.1 Numeric Results

The table below describes the numeric results of our implementation, including both the manual 'actual' count and the counted number by the program.

Clip	Fruit	Quantity	
		Actual	Counted
1	Banana	10	9
	Orange	0	0
	Tomato	0	0
	Green Apple	0	0
	Pear	0	0
	Other	0	0
2	Banana	0	0
	Orange	3	3
	Tomato	12	12
	Green Apple	0	0
	Pear	4	4
	Other	3	3
3	Banana	0	0
	Orange	7	7
	Tomato	12	12
	Green Apple	0	0
	Pear	0	0
	Other	1	1

Clip	Fruit	Quantity	
		Actual	Counted
4	Banana	0	0
	Orange	0	0
	Tomato	0	0
	Green Apple	3	0
	Pear	5	8
	Other	0	0
5	Banana	0	2
	Orange	0	1
	Tomato	7	7
	Green Apple	9	0
	Pear	10	12
	Other	0	7
6	Banana	52	57
	Orange	0	4
	Tomato	0	0
	Green Apple	36	0
	Pear	61	87
	Other	0	37

3.2 Accuracy

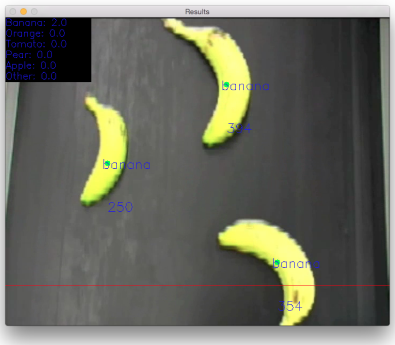
The table below describes the accuracy of our implementation with regards to number of fruit in the video clip:

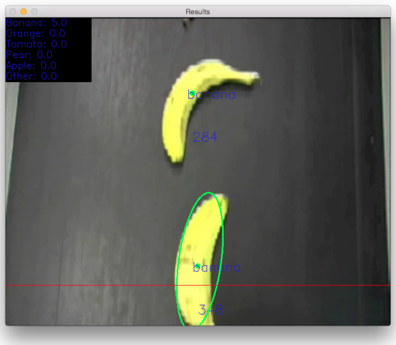
clip	1	2	3	4	5	6
accuracy	90	100	100	100	61	69

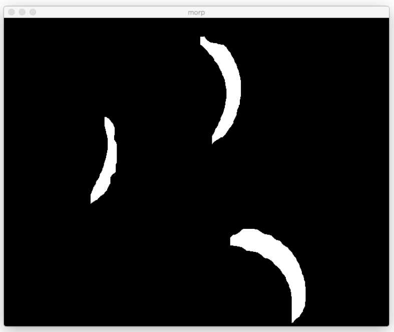
3.3 Example Images

The images shown in the following section are screen shots from the running program. The left image is the standard screen shown when running. The black box contains information about the current real time count, and the labels on the image signal what the fruit is currently being classified as. The image in the middle of the set contain contours drawn on the image. These are not always accurate as the contours are drawn based on the original binary image rather than the image where morphology is applied. The right image is the morphology version of the top left image.

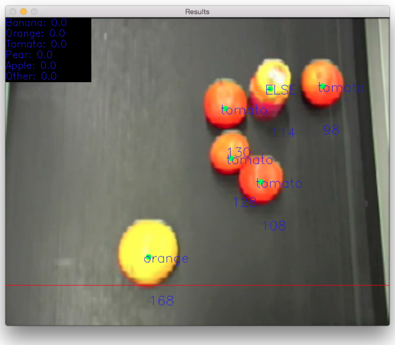
Banana

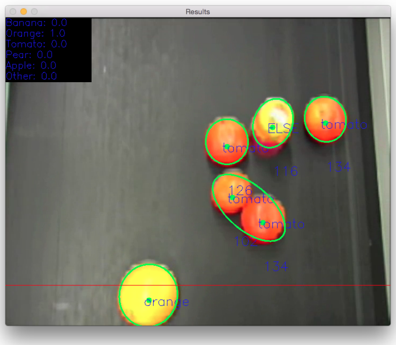

```
Banana: 2.0
Orange: 0.0
Tomato: 0.0
Pear: 0.0
Apple: 0.0
Other: 0.0
```

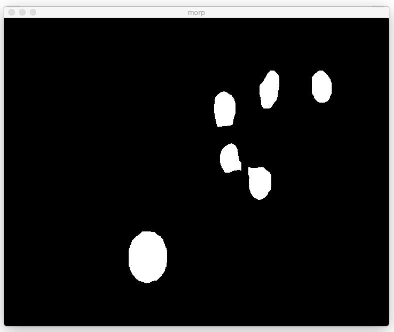

```
Banana: 5.0
Orange: 0.0
Tomato: 0.0
Pear: 0.0
Apple: 0.0
Other: 0.0
```



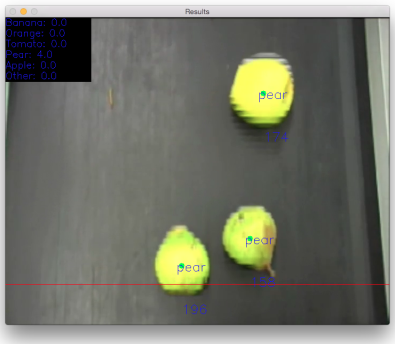
Orange, Tomato and Other

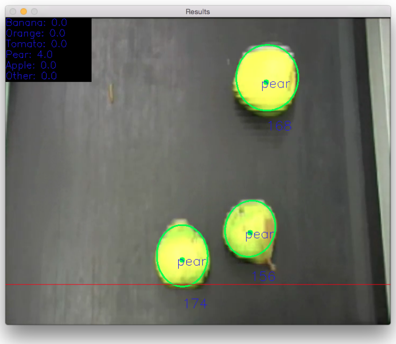

```
Banana: 0.0
Orange: 0.0
Tomato: 0.0
Pear: 0.0
Apple: 0.0
Other: 0.0
```

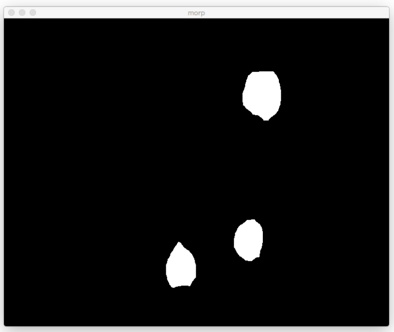

```
Banana: 0.0
Orange: 1.0
Tomato: 0.0
Pear: 0.0
Apple: 0.0
Other: 0.0
```



Green Apple and Pear







4 Discussion, Observations and Conclusion

One of our initial implementations of this project was completed in Matlab. This implementation had many issues when applying the required preprocessing functions which causes the video to run at 1 frame per 1-2 seconds. This version had a very preliminary count of bananas, oranges and tomatoes. This was one of the major reasons which made us decide to implement this in another programming language. This implementation was also before we discovered what the watershed algorithm was, which significantly improved morphology speeds in our final implementation. This leads us to believe that matlab's implementations of morphology are flawed and slow compared to the opencv libraries in python or C.

In the video clips provided, a lot of issues were based around objects touching, and consequently a lot of time was devoted to dealing with this issue. A lot of trial and error was done using basic morphology functions to attempt to get this working, however this always seemed to cause more problems than solutions. Particularly this was the case because if it worked for all fruit except bananas, then bananas were nearly always missed due to being eroded far too much. The opposite case where bananas were perfectly dealt with caused problems with groups of touching fruit not separating. Due to frustration, this is where the idea to subtract the first frame of the video from every other frame to reduce processing time and noise came from. Through further research this is how we discovered the watershed filtering method on the openCV help and tutorial pages.

The major problems encounted were in the seperation of fruit which were touching. Either an algorithm would be too computationaly expensive or would not add too much complexity to the code base. Examples of these methods include comparing fruit to known contours of the fruit we had selected and canny edge detection. In the end we settled on the use of the watershed algorithm which seems to work well. For classification we attempted to solve the problem using an ML perspective and classified by a weighted sum of the features. But we found this method too complex to code by hand and we do not believe we have enough training data to use a more complex approach (such as a SVM) as it would overfit the data. In the end we hard coded boundaries for the fruit which they were allowed to fall under and then ran a decision tree in order to classify the fruit. This approach is very inflexable and given non-toy example we would have implemented a ML based approach for the classification.

Our implementation works perfectly in the first four clips provided, and after this become highly inaccurate. We believe the change in accuracy between these sets of videos to be due to changes in lighting. In a real world implementation of a project like this it would need to be able to deal with different lightings as lighting is effected by time of day and weather. As the way we were classifying the fruit was naively done using only colour and basic size measurements of the fruit, potentially the best way to deal with the change of lighting would be to implement a better way to classify the fruit. In research papers about this topic, presented earlier in the paper, many suggested using photos and averages of the groups of photos to deal with different conditions, in addition to a more complex method for separating fruit types than a decision tree.

One of the biggest issues which we didn't solve was how to classify the green apples and pears as separate types of fruit. One of the initial ways we believed would allow us to classify the two as different was the shapes of each fruit. What we didn't account for is that when a pear is sitting on its base, from a top-down view it looked the same shape as an apple. Additionally after applying the watershed filter the shapes of the fruit were lost. When dealing with all the information gained from the contours function, the average for these two objects came out near identical. This meant that the way to date that we classified for fruit could not differentiate the fruit. The solution we suggest to test next is the same as mentioned in the paragraph above to deal with different lighting conditions.

The implementation did not deal with the condition where fruit rested upon each other. This condition was not possible to deal with in the current implementation as the only fruit put in this condition was a banana, and we measured bananas by extent. Extent is a number between 0 and 1 which represents how much of the area in the bounding box drawn around the object is actually the object. Because of the curve of bananas, they never take up more than half of the bounding box at most. This value of the extent generally did not change when there were multiple bananas in the box. A potential solution to this would have been to use the areas, and based on the area of the pixels, take an educated guess as to how many bananas were there. This most likely would have worked for two bananas, but not the situation where there were three bananas. The more complex approach we were considering implementing but time stopped us trying was to match shapes and to count based on the number of shapes in the object.

4.1 Future Work

Further work in this field should be done by researching into :

- identifying fruit resting on top of each other
- ways of classifying the difference between pears and green apples
- handing different lighting situations

4.2 Conclusion

1. Basic shape and contour information of an object can successfully provide a preliminary object classification to rule out the cases which the object is obviously not.
2. Further, more complex classification methods are required to accurately classify and count fruit reliably.

5 Learning Outcomes

This project has allowed all members of the team to explore new ideas, to cement concepts learnt in the course as well as develop a general understanding and feel of computer vision. The following points really stood out:

- Computer vision is highly domain specific where solutions need to be tailored to particular problems rather than generalised. We spent a lot of time working towards clasification of fruit generally through means such as simplistic neural networks before we defaulted to decision trees as they were the most reliable.
- The watershed method is a good method for separating items which are touching, but can cause issues when there is noise present.
- The soft minimum thresholding approach is more effective for removing general noise while not removing other details of the image like applying a binary threshold does.

6 Using the Source Code

To run the code, navigate to the `source_code` folder in terminal. Once achieved, type the following command: `python final.py path/to/clip`

System Requirements: Python 2.7, `cv2` python library, `numpy` python library

6.1 Commands

There are built in commands for our implementation. These work by pressing any of the keys mentioned in the table below once, to get the intended effect. With regards to the keys corresponding to frame rates, multiple key presses of the same key will work, however holding down the key will not.

Key	Function
P	pause/resume the video
C	toggle the count display
L	display/hide labels of objects
Q	quit
B	display/hide the outlines of the objects
S	slow down the frame rate
F	increase the frame rate