Embedded Software Engineering

Assignment #1

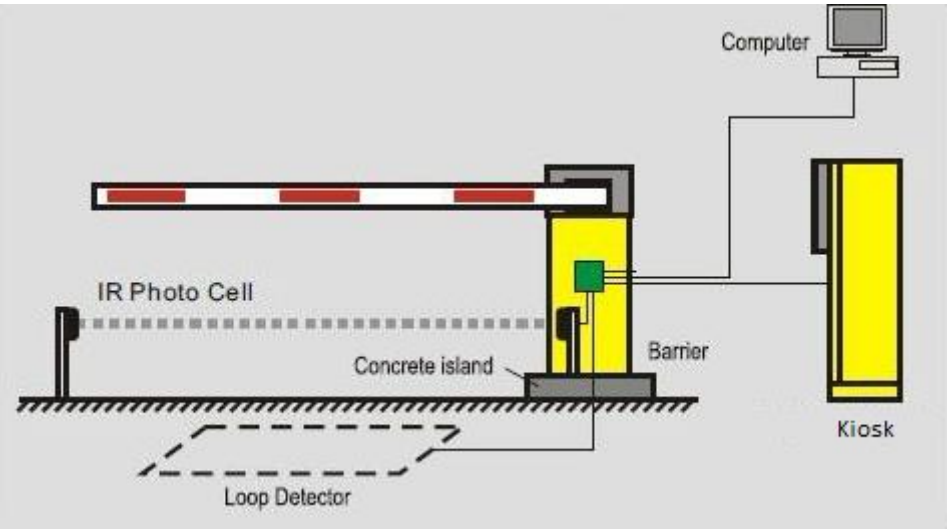
This assignment is part of the overall course assessment. The following general points apply to all work handed in for assessment:

- The assignment should be completed on a word processor such as Microsoft Word with suitably commented Python code included.
- Every assignment must have an AUT cover sheet signed and dated by you, and stapled to the front.
- You are allowed to discuss the problems with your colleagues, but you must hand in your own work.

---

5 Marks

1. You are to design a software solution for the following problem. Your solution should include relevant UML diagrams, such as class, sequence, activity and/or state, but not code. This is a design exercise in conceptual modelling, so review of lectures may be useful.



The above diagram illustrates a standard parking building automatic barrier arm at the *entry*. It consists of:

- An automated barrier arm (raise/lower)
- An IR beam for detecting if an object is under the arm (open/closed)
- A loop detector for detecting if a car is in front of the arm (empty/detected)
- A parking kiosk with a button a user may press to be issued a parking ticket to enter the car park
- A connection to a central computer where the control algorithm will run, which also logs the time when a car enters the car park in a central database.

Use your knowledge of how a typical car park entry barrier arm operates (or research it if you do not know how they work) to develop an operating strategy for the above configuration, and then develop an object-orientated solution using UML models.

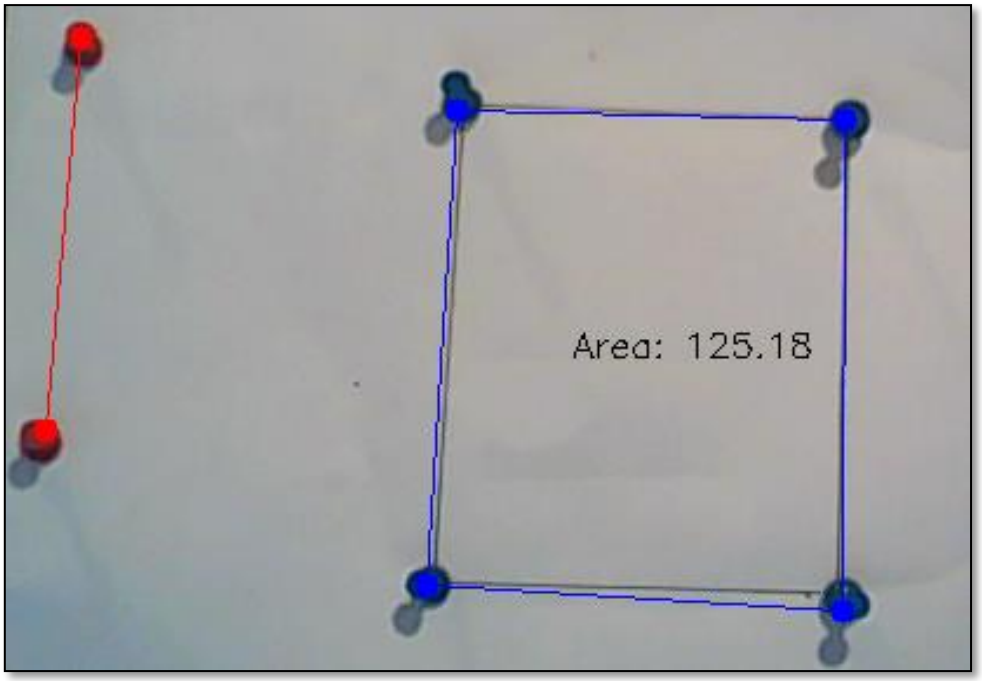
2. You are to develop an object-orientated 2D area estimator using Python, OpenCV and the USB webcam you have been issued. Your application may run on the Raspberry Pi (preferable), or you may use a desktop PC.

10 Marks

The operation of the area estimator is as follows:

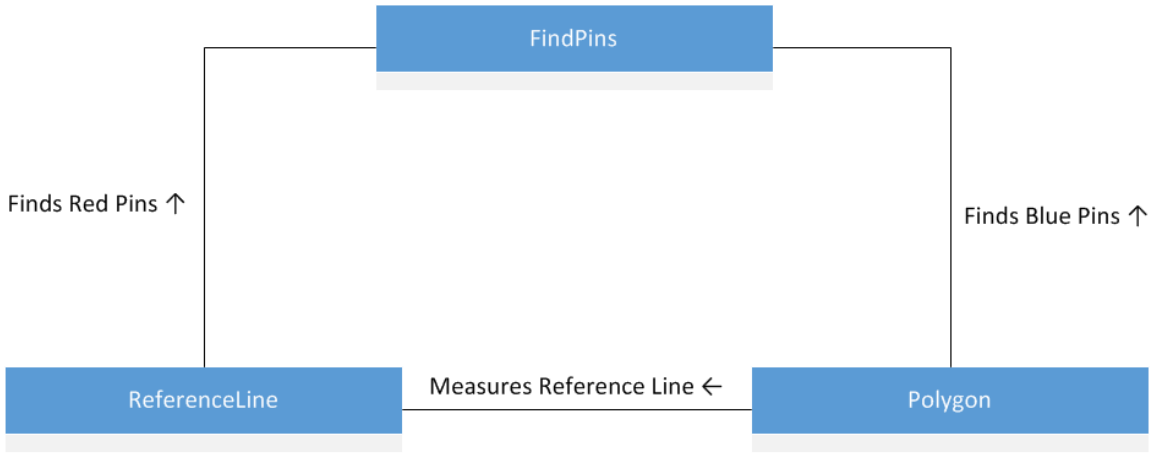
1) Two red pins are placed 100mm apart within the target frame (i.e. a blank notepad or other uniformly coloured surface). These represent a reference measurement for the vision algorithm.
2) Three, four, five or six blue pins are placed in a closed polygon in the target frame, representing the vertices of a shape to calculate the area of. The 2D shape the pinned polygon represents does not have to be regular (i.e. each side can be a different length), but it must be convex.
3) The webcam is placed vertically above target frame with good lighting.
4) Your algorithm must then iteratively:
    a. Read a frame from the webcam.
    b. Identify and mark both red pins. This require you to use some of the image processing techniques you have been learning, such as filtering, thresholding, contour identification and finding centroids.
    c. Draw a line between both red pins, then calculate the length of the line in pixels.
    d. Identify and mark all blue pins.
    e. Using the OpenCV function `convexHull`, find a convex polygon which includes all blue pins as its vertices.
    f. Draw lines between each of the vertices identified with `convexHull`.
    g. Using the OpenCV function `contourArea`, together with the reference 100mm line as a scaling factor, calculate the area of the polygon in real units (i.e. mm^2 or cm^2), and display it within the polygon.

A sample output is shown below. The true rectangle area was $120cm^2$, thus the estimate is reasonable.



Software Design

You are to use the following software architecture to write your Python application:



`FindPins` Class:
Applies the image processing techniques you have learned to identify a number of pins of a selected colour.

`ReferenceLine` Class:
Uses a `FindPins` object to determine the location of the reference line, then calculates an appropriate scaling factor for use in area calculations.

`Polygon` Class:
Uses a (second) `FindPins` object to determine the location of the pin vertices, then applies the convex hull calculation to determine the area, using the `ReferenceLine` object to determine the required scaling factor.

Your main function should instantiate the required object(s) and then repeatedly execute the algorithm to provide 'real time' 2D area estimation. Obviously your code should be professional written, with proper encapsulation, exception handling, structure, commenting and result presentation. For example, you may like to apply a moving average filter to the area calculation to reduce noise and spurious pin identification effect.

Assignment Write Up

Ensure you test at least 5 different polygons with a known area with your algorithm, including screenshots (and/or YouTube videos) of the results in your assignment. Comment on the robustness of your algorithm, and identify any weaknesses (such as lighting, orientation, filtering, etc).

In addition, your write up must contain a UML class diagram for each class, as well as a UML sequence diagram showing the operation of your program.

Hints

- Start by developing an algorithm to robustly identify the location of pins. This will be the most difficult part of this project.
- Try keeping light intensity and camera position constant.
- Use HSV rather than BGR for matching colours.
- Feel free to use other coloured pins. You may find other colours are easier to isolate.
- You will need to get familiar with manipulating and indexing NumPy arrays. The following document is a good start:
    - http://wiki.scipy.org/Tentative_NumPy_Tutorial