Criterion A: Planning

Definition of the problem and justification
I will be my own client. There is no real problem that is being solved; it's more a theoretical problem. This problem is the ability to train a program to find objects in images. The idea for this, to find objects in images, is not a new idea, nor is the method.

To create this project I have selected to use the OpenCV library with c++ because OpenCV is a very robust and useful method of image processing and analysis. I used c++ because I have the most experience using c++ with the OpenCV libraries.

Words 95

The proposed product
The OpenCV Libraries offer a matrix/image codex that is very good at storing image data and doing mathematical operations with them. OpenCV also had a lot of build-in methods such as convolutional template matching and getting the Sobel images. This along with other reasons is why OpenCV is used to do this. The reason c++ was used is because of how versatile and useful the language is, also because I know how it works with OpenCV. I'm using Visual studios because it supports c++ 14 and because OpenCV was made to work best with visual studios. I'm used visual studios 2017, and while I did not use it, the community edition for visual studios is free and works with OpenCV.

The Scenario For this project is how to find objects in images using pre-generated keywords. This program works in two parts
Trainer: This function will take an input of multiple similar images of the object in the same scenario. The program starts with the first image and splits it into boxes of features (using a gradient image to find these boxes). It then repeats this step for all of the other images giving a lot of features of the images. Then it looks through each of the images to find if any of the features are similar and then adds those to a list in a stack (although I'm just going to use a list). When finding the similarity the scale of the "Keyword" is varied as to compensate for different image sizes. When the lists are made each image is combined with each other image at the point where the best correlation is relative to the first image. This will make a "blurred" image ready for use as a "keyword." The range for the vector is given by the range at which each "keyword" lies within the image.

Detector: My first project uses edges to find circles and lines in an image. Instead of using edges this program will use common image samples (the "keywords") to identify where and what the object is. Each "keyword" will be like a blurry key portion of the full image as to aid the correlation process. The "keywords" are stored in a database along with what it represents and a vector that points to the center of the object. Using this technique, the program finds the images equivalent (all that pass a correlation threshold) of what the "keyword" represents. This is done for every "keyword" and with variations like rotation and scale. When each "keyword" is correlated the vector is read, and the possible object location is given from the image's location

plus the vector plus the vector. It is important to note that each vector in the list is a really a range so that the program can account for errors in scaling and other discrete variation.

Words 476

Success criteria

- Have a Program capable of creating “keywords”
- These keywords must have vectors pointing to the center of the object and the keyword itself
- Having the ability to store and retrieve these keywords
- Have a program that can use the keywords to find the object
- Indicate where the object is

Words 49

Total words 640