

Southeastern Louisiana University – CSIT Department  
CMPS 441 Artificial Intelligence  
Midterm Report – Spring 2017

# First Person Shooter AI with Computer Vision

Yashmin Sainju, Brian Martinez, Justin McLin, Subash Mishra

## Introduction

### Motivation

The idea behind the system is to provide an AI that can navigate itself through a randomly generated environment, while avoiding enemy traps and consequently defeating the enemy.

The main area of research of this project involves computer vision alongside machine learning and implementation of genetic algorithms that will strive to be an improvement on current models that solve similar problems. The motivation behind implementing computer vision to this game instead of real environment is that the game is bound to have constrained environment and so the deficiencies of cameras will no longer be an issue. Also, machine learning will require large datasets, and games can provide consistent and easily reproduce-able environments. With the rise of powerful hardware and better algorithms, both the machine learning and the computer vision fields have seen tremendous growth.

### Problem

Unlike current implementations of AI in games, which interact with the game through an application programming interface (API), this system will strive to interact with the game based on information provided to it through computer vision.

### Challenges

In almost all available gaming AIs, the algorithms that have been used for them to achieve similar tasks have been implemented with the AI being omniscient. This system will completely remove that aspect and instead will only be able to take decisions by using the information it gains through its vision. As there is very little available research in this area, there are not many comparisons that can be made. The idea of performance must be taken into consideration. The best performing algorithms in terms of accuracy may also be one of the most time-consuming algorithms. Since the game is going to be in real time, unlike turn based games, reaction speed must be at its optimum.

### Objective

The high-level objective for this project is to create an agent that can visually enable an AI to gain an understanding of the environment it is in, recognize the objects around it, and therefore, interact accordingly with them. The AI getting the aid will be completely blind to the environment except for the information provided by the system. While the system algorithm will be independent, the AI will be run on Unity3D platform and the system will be implemented on OpenCV python.

## Related Work

### Key Concepts

Following concepts are the topics of interest in processed\_img(original\_image) and haar\_cascade(gray).

2.1.1 Object Detection

Object detection is the process of detecting instances of objects of a certain class in an image. The application will require a strong implementation of object detection. This will be the basis of recognizing and distinguishing objects in the game. These methods will vary in efficiencies across 2D and 3D games. However, the application will try to develop a generic technique which would work well on both these platforms. The following will be the criteria for the techniques to be evaluated:

Recall

This will give us the ratio of correct positives of the objects that are detected.

Number of correct positives

Total number of positives in dataset

Accuracy

This will give us the number of false detections made on the object.

Number of correct positives

Total number of positives + Total false positives

Training Speed

The speed with which a classifier learns from the sample set.

Detection Speed

The speed with which a technique can find specified objects in an image. The faster the detection speed, the more video frames it can analyze.

2.1.2 Template Matching

This is the simplest and the most potential technique for object recognition for 2D images. An image of an object (a template) is slid over the input image at each point calculating the accuracy of the match. Although the method to calculate the accuracy of the match may vary, they all involve cross-correlating the corresponding pixels. This method can produce very accurate matches if the template to be matched is virtually identical to the objects in the input image. This is true for most 2D games since the images, which may be static or moving, are always identical.



*Fig 2.1.2 Template Matching*

Although this technique is translation invariant, it is not invariant to scale and rotation. Performance is another issue since an entire image must be iterated through. An image of resolution XxY and a template of resolution MxN gives a complexity of O((X-M)\*(Y-N)\*M\*N) considering the pixels near borders that will not need to be evaluated.

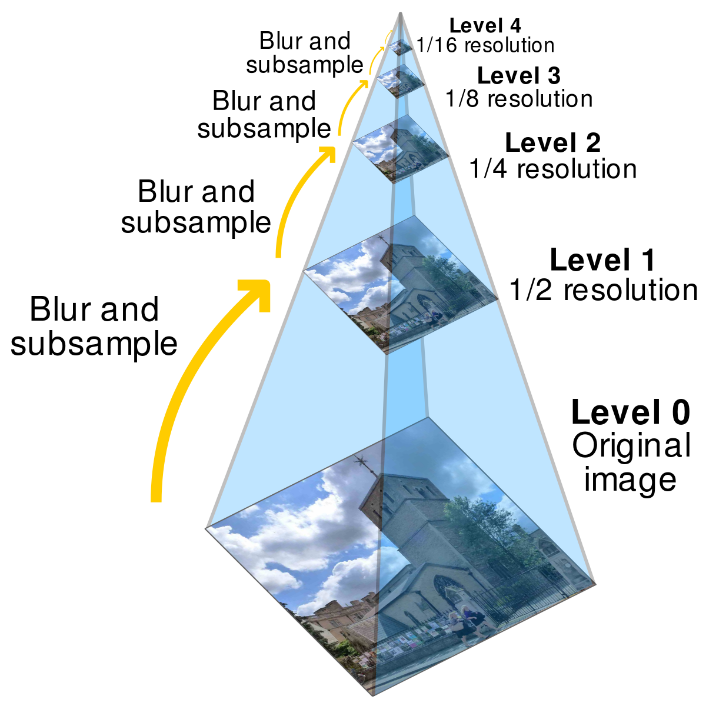
Implementation

The implementation for template matching that the application uses is the *matchTemplate()* implementation from OpenCV. The Open source Computer Vision is a library of programming functions with a strong focus on real time computer vision.

The implementation does matching in the frequency domain instead of spatial domain as correlation is much faster in the frequency domain. Fourier transforms of both the template and the image are taken. To multiply the two images in the frequency domain, they must be the same size, and so the template is padded with black pixels (zero padding). This results in only O(N2logN) operations for NxN images.

2.1.3 Image Pyramids

An image pyramid is a data structure that has found wide applications in a wide variety of vision agents. A sequence of copies of the original image are successively down-sampled in regular steps.



There are two types of image pyramids: Gaussian pyramid and Laplacian pyramid.

Gaussian Pyramid

The Gaussian Pyramid is used to filter and sub sample the images repeatedly to generate the sequence of reduced resolution images. G0 is the original image, at the bottom of the image pyramid. To produce level Gi + 1 from Gi , we first convolve Gi with a Gaussian kernel and then remove every even-numbered row and column thereby producing an image 1/4 of the area.

Laplacian Pyramid

The Laplacian pyramid is required when an up-sampled image must be reconstructed to an image lower in the pyramid. For instance, converting a G1 to a G0. Unlike Gaussian pyramids, Laplacian pyramids are complete image representation. The steps used to construct the pyramid may be reversed to recover the original image exactly. Such constructions tend to enhance image features like edges. Pyramid representations can also compress data.

Pyramid methods provide a way to execute some level of template-matching scale independence. They are also used to estimate properties within local image regions. Initial segmentations are fast and are done on low resolution images in the pyramid and then are refined on subsequent levels.

Laplacian pyramids will be used in the application as a technique for image enhancement such as noise reduction although images from games are perfect and noise free.

2.1.4 Background Subtraction

Background Subtraction is a way of detecting moving objects from the rest of the image. One can either run object detection of the entire game “screenshot” or just use it with a good background subtraction algorithm. While the application will still need to detect static objects often, detecting enemy AI would use background subtraction. There is a good possibility that it will lead to a higher FPS. The question is if it will be as accurate.

The simplest background subtraction methods just subtract one frame from another and then any difference past a certain threshold is classified as foreground.

*|framei – framei-1| > threshold*

A model of the background should be learnt first so that it could be compared against the current frame and so the backgrounds can be subtracted. The things that remain are foreground object – usually moving.

However, more complicated algorithms must be used in this background subtraction since the background objects may be moving too and illumination changes may affect the results. Techniques that will average the differences of the pixels of the scene, based on a Gaussian distribution would be more effective.

This will not be much of a concern for 2D games; however, for 3D games, with advanced lightning, the background model will have to be kept regularly updated to adapt to various geometry settings and luminance conditions.

2.1.5 Gaussian Mixtures

The major problem with frame subtraction for background extraction as described above is that it cannot be generalized for 3D games. The background model has to be learnt for effective background models in an environment where there is periodic movement and changing lighting conditions. Hence, the generalized mixture of Gaussians (MOG2) implementation provided by the OpenCV library. Mixture of K Gaussians (αi , σi , ωi) is an advancement on the running Gaussian for multi-modal backgrounds. αi denotes the length of history, σi is a parameter for threshold, and ωi is a Boolean for shadow detection. This method models both foreground and the background. To be able to distinguish between which distributions only model the background, all the distributions are ranked according to ωi σi and the first X are chosen to represent background. One could also use distributions with a ratio above some threshold instead of the ranked distributions.

2.1.6 Edge Detection

Edges occur where there is a significant change in image intensity between two different regions in an image. It examines the rate of change of intensity near the pixel: sharp changes with steep gradients are evaluated as an edge while slow changes are neglected.

Edges can be classified as step edges, line edges and roof edges. An edge detection algorithm produces a set of edge points from an image. These involve convolution masks to find edges in vertical, horizontal and diagonal directions. In the project we will be using an interesting edge detector as proposed by J.Canny.

The canny edge detector tries to assemble individual edge candidate pixels into complete contours. There are two main steps:

First Derivatives: The first derivatives are computed in x and y and then combined into four direction derivatives. Points where these directional derivatives are local maxima are candidates for assembling into edges.

Contour Construction: Canny uses thresholding with hysteresis. This requires high and low thresholds. The following rules are applied to each pixel:

* If the gradient of a pixel is higher than threshold, it is accepted.
* If the gradient of a pixel is lower than threshold, it is rejected.
* If the gradient of a pixel is between high and low thresholds, it is accepted only if it is connected to a pixel that is above the high threshold.

This algorithm hence, traces object boundaries with more accuracy compared to other operators since it is not affected as much by noise and blurriness in images.

2.1.8 Haar-Like Cascade Training

Haar Cascades is a technique of training a program to recognize a specific object or kind of object in the given game. This is done by providing the program with a number of “positive” images and a number of ‘negative’ images. Preferably, the size each individual negative image more would be bigger than that of positive images. The positive image is then superimposed on each of the negative images creating a number of ‘positives’. Finally, the AI is ‘trained’ to recognize the positive images inside a negative background. The more the number of negatives, the better the chance of object recognition gets.

## Architecture

### 3.1 Requirements

The following are the use cases of the project:

* identify walls, traps and any other object in the game in real time.
* identify enemy.
* have an understanding of the game environment.
* recognize change of environment and related events in the game.
* recognize patterns that might help the actor gain a better understanding of its situation throughout the game.

3.2 Overview of System Architecture

All the computer vision and machine learning algorithms will be implemented in python and reside in a collection of classes encompassed by the ‘Core Vision’ node. Any program intending to use the application will need to make a function call to the application and the application will respond by sending either a Boolean value, an (x,y) coordinate or an array of coordinates back to the program. There will be only 1 class of functions in the application: call-ins.

Call-ins: These will be vision call ins which interact with the vision core of the application such as vision.getEnemyCoordinate() and vision.isEnemy(). Although the application will be monitoring the program screen all the time, it is only upon receiving these call-in requests that it will evaluate the situation and output respective values.

There will also be some unidirectional information sharing at the startup, where the actor will be able to assign certain configuration parameters:

optimumFPS = 30

zeroCoordinate = (100, 100)

screenSize = (500,600)

This script will initialize various variables on the application if specified by the actor. Otherwise, the application will run with its default values. Although more access could be granted to the vision application, it would be best to make it simple and easy to use.

3.3 Platforms

The development platform for this application will be a Windows system. To be able to interact with screens within Windows, the application will need to be able to interact with the Windows GUIs. This will be by implementing the above-mentioned interfaces and method calls.

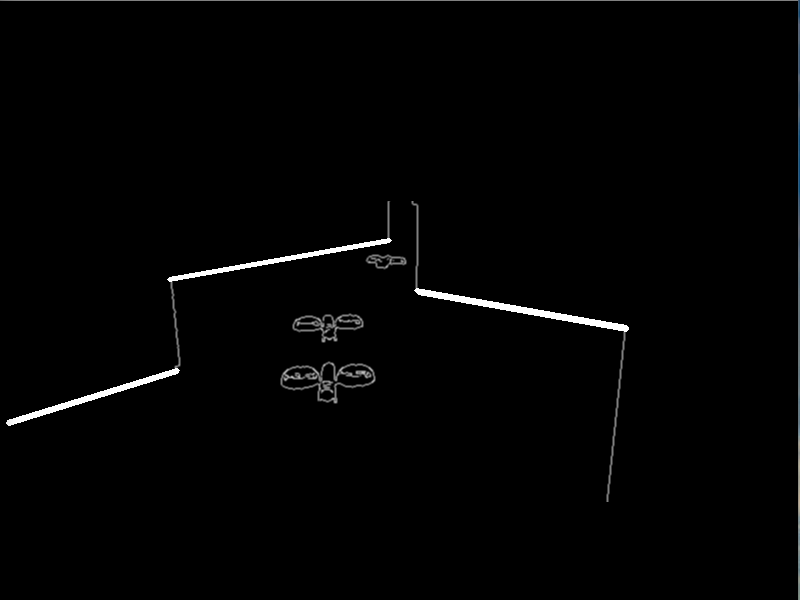
## 4. Evaluation

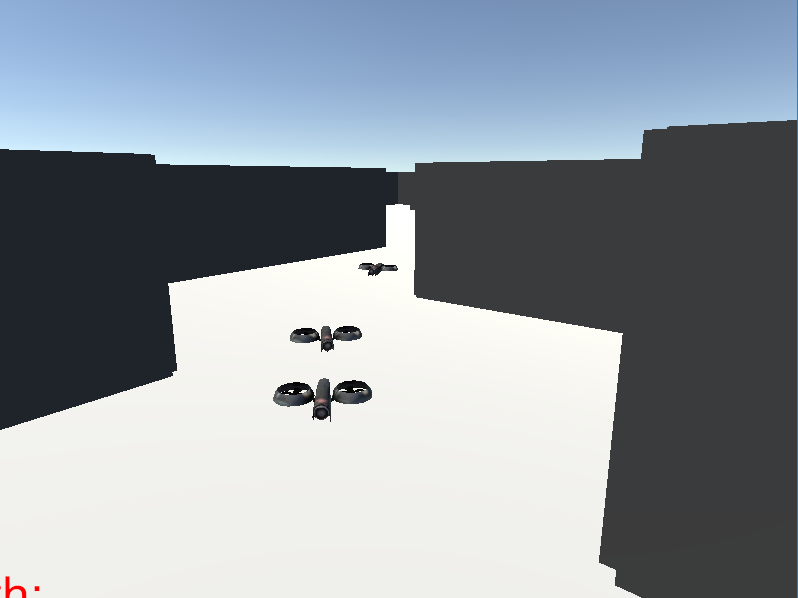
4.1. Implementation

The following were implemented in the project to assist computer vision:

Implementation and testing was done with dependencies on OpenCV 2.7.5 and python 2.7.13.

Edge Detection

 Edge detection was done using canny algorithms as supported by OpenCV. The two thresholds were optimal at 200 and 300 respectively. The accuracy was 100%. However, detecting edges was not sufficient hence, Gaussian blur was used to smooth the lines out. Next, regions of interest were defined in certain vertices of the 800x600 screen which allowed the use of Hough Lines that would help in background extraction and Haar Cascade training. Overall, the accuracy was 100% and the speed was ranging at the max of 11 FPS and a min of 6.7 FPS.



*Fig 4.1.1 Real Frame Vs Canny Edge Detected frame with Hough lines in regions of interest*

Background Extraction

The background was extracted only on the area of interest as determined by the edge detection. This enabled the software to only extract within the Hough lines as created by the edge detection algorithm and hence, not have to scan the pixels in the entire frame. A minimum in speed anywhere between 8.13 FPS and 10 FPS along with the edge detection algorithm. Without the edge detection, the background extraction ran between 11 FPS to 15 FPS.

Haar Cascade Training

The training was done with OpenCV and python on Ubuntu 16.04.2. After the training and generation of cascade xml files, however, the software was run on Windows 10. At the beginning, there were 7 negatives used with only 1 positive image, which were later superimposed using opencv\_createsamples to create 15 positive image files that were used to train the cascade classifier in 4 stages. This was done for three separate groups of objects, i.e. enemies to be identified. Later, it was run both with and without background extraction and edge detection algorithms. The following were the results:

Implementation with canny edge detection: The implementation was between 5 FPS to 8 FPS. The process was relatively slow; however, the detection speed was much higher. The recall was about 99% accurate for drones and 86% for warriors. However, the accuracy was extremely low. This probably was caused by the sample positive images that were fed to the cascade. The images were neither vector, nor set on a realistic background in terms of the virtual world, i.e. the background on them were black, and these images were as small as 30x30. Hence, multiple black spots were marked as drones.

Implementation without canny edge detection: The implementation ran anywhere between 10 FPS to 15 FPS. And the recall was 99% for drones and 86% for warriors as with edge detection. However, the accuracy was twice as low as with edge detection because of the same regions, plus the added pixels of the entire frame without separation of region of interest.

## Conclusion

The agent so developed met a larger subset of the requirements presented in the project. While some algorithms performed the best with speed, some others were better with accuracy. However, since no performance lag was detected in the actual game, an algorithm with higher accuracy at the expense of reasonable speed would be optimal for a project like this. Also, there are huge prospects of attaining this goal using GPU implementation of the same or better algorithms. A large set of positive and negative images for classifier trainings would make the object detection more accurate without compromising speed.

# Deliverables

https://github.com/YashminSainju/eyesforai

Bios

* **Student Yashmin Sainju** – Sainju is a junior Computer Science student in the Computer Science and Industrial Technology Department at the Southeastern Louisiana University. She has completed relevant courses and is expecting to graduate with a BS degree in Fall, 2017.
* **Dr. Wesley Deneke, Mentor** – Deneke is a professor in the Computer Science and Industrial Technology Department at Southeastern Louisiana University. He leads student research projects that are currently focusing on how to simulate Human Workflows using 3D virtual worlds.

Bibliography

Shivani Agarwal and Dan Roth. Learning a sparse representation for object detection.

In Anders Heyden, Gunnar Sparr, Mads Nielsen, and Peter Johansen, editors, *Computer*

*Vision* ECCV 2002, volume 2353 of *Lecture Notes in Computer Science*, pages 97–101.

Springer Berlin / Heidelberg, 2006.

Murat Kunt. Digital image processing. In Dionys Baeriswyl, Michel Droz, Andreas

Malaspinas, and Piero Martinoli, editors*, Physics in Living Matter*, volume 284 of *Lecture*

*Notes in Physics*, pages 73–75. Springer Berlin / Heidelberg, 1987. 10.1007/BFb0009210.

P Aschwanden and W Guggenbhl. *Experimental Results from a Comparative Study on*

*Correlation-type Registration Algorithms*. Wichmann, 1992.