Research documentation (In progress)

Aims

To integrate a computer vision library (Original idea is OpenCV) into Unity and use the library to track facial features. This project has a heavy emphasis on finding ways to accurately detect jaw movement.

Initial thoughts and observations

OpenCV is a C++ library and to integrate the library with Unity I would need to find a plugin or find an external wrapper that would be compatible with OpenCV.

Optimisation is key as the tracking will need to be done at least 30 times a second (webcam minimum 30FPS). This limitation means there is constraints on what type of tracking can be done.

I am going to need a further understanding on how facial tracking works in order work out how to track jaw movement. An initial thought was using edge detection and location of eyes and mouth.

Research

In this part I will be talking about what I have researched, what issues I have discovered and what I would recommend to do next.

As I initially noted OpenCV is not compatible with Unity, researching on what other alternatives are available I have found the following:

1. [OpenCVSharp](https://github.com/shimat/opencvsharp) – This is frequently updated and there enough documentation/guides available to help you understand it. It also has Visual Studio auto suggestions to help you.
2. [Emgu CV](http://www.emgu.com/wiki/index.php/Main_Page) – Does support Unity, it cross platform and apparently is more efficient than OpenCVSharp.
3. [OpenCV for Unity](https://www.assetstore.unity3d.com/en/#!/content/21088) – Available from the plugin store, saves time with installation but is costly. User reviews did not look promising so I would avoid this option. However, I found the GitHub to this plugin [here](https://github.com/EnoxSoftware/OpenCVForUnity). Unsure how you would install it if you downloaded it from here.
4. If you are looking to develop for ARM-based processors then fastCV could be an option, has its own SDK and has strong documentation. I have not considered this as much as the aim of the project is to try and keep it in Unity.

Augmented Reality?

As I was unsure if augmented reality was a part of the project I did a little research to see if there are any tutorials/ plugins that deals with facial recognition and augmented reality in Unity.

Most of the research ended up with commercial SDKs which would not be suitable for the project however I discovered a couple which may be worthwhile to look at in the future:

* [Eyeris EmoVu](http://emovu.com/e/) – Need to request SDK from company, not sure if it is free. It is well documented and furthermore there are 4 different types of face analysis. These include:

1. Face detection.
2. Emotion recognition.
3. Age recognition.
4. Gender recognition  
   There is also a facial tracker feature which creates bounding boxes and can detect the head pose estimation.

* [Vuforia](https://developer.vuforia.com/) – This AR software is partnered with Unity so it is easy to create an AR project with this. However, this does not include facial recognition therefor, the OpenCV wrapper will still be required. Unfortunately, there are licensing costs, but while the app is in development it does not cost. There are limitations on the free development version.
* <http://www.justapixel.co.uk/how-to-make-an-ar-app-in-5-minutes-with-unity-and-vuforia/>
* <https://library.vuforia.com/articles/Solution/Compiling-a-Simple-Unity-Project>

Other Libraries

* [Dlib](http://dlib.net/) improves upon OpenCV and gives better performance. It also provides HOG (Histogram of oriented gradients) as well as basic haar cascades. I found a C# wrapper to this in the RageAppliedGame repository.   
  If Unreal Engine is used I would recommend using this library instead of OpenCV.

Game engine suggestion

[Unreal engine](https://www.unrealengine.com/what-is-unreal-engine-4) – I recommend using this game engine over Unity as it uses C++ for its scripts. This means you can integrate the default OpenCV library into it without the wrapper. This would provide a massive improvement on performance. It also means other libraries that extend from OpenCV, like DLib (will discuss more later) could also be used. Just like Unity it comes with strong documentation and an active forum community if you need help. There are also plenty of video guides to help introduce you to the engine.

What I chose

To attempt this project, I decided to go with OpenCVSharp as it was the first wrapper that I found and I already found a demo project that uses Unity [here](https://github.com/qian256/OpenCVSharp3Unity). This gave me a starting point and helped me to understand how to use the library. An additional benefit I found with this project is that it demonstrated a way to convert a Mat object (which is what OpenCV uses for images) into a Texture2D object, which then can be textured onto objects onto Unity. An issue that I noticed from using this software was how slow the conversion was, this meant that there was a noticeable delay between the camera and what was shown on screen.

For the purposes of this project I kept with using the standard window, and primitive shape builder that OpenCVSharp provided, I noticed that there was almost no delay between the camera and the display. However, if you wanted to create textures that would be used in OpenCV I would look to see if there is a way to improve the conversion.

Installation

To install OpenCVSharp into Unity use the following steps:

1. Create empty Unity project.
2. Open project solution folder in Visual Studio.
3. Go to Tools-> NuGet packet Manager.
4. Open manage packages for solution.
5. Search for ‘OpenCVSharp3- AnyCPU’ by Shimat.
6. Install package.
7. Project should be successful (still need to work out how to build)

Once these steps have been followed you should have something that looks like this:

Todo: Insert pictures

If you are still unsure you can follow the guide [here](https://github.com/shimat/opencvsharp/wiki).

Proof of concept

As a proof of concept, I did a basic face detection to start learning how facial recognition worked.

The algorithm for this is as follows:

1. Load Information (webcam location, XML files to test against)
2. Capture frame from camera.
3. GrayScale captured frame.
4. Equalise histogram to help detection.
5. Run CascadeClassifier.DetectMultiScale() function.   
   This function take the following parameters.  
   - Mat image, this is the image that has had the previous steps applied to it.  
   - Float scaleFactor, this is the size that the image is scaled reduced by each scaling.  
   - Enum HaarDetectionType, applies certain flags to the cascade.  
   - Size (x,y), size of the rectangle that will be returned.
6. For each face returned; draw a rectangle around feature.
7. Display frame with rectangles in separate window.

Results

From this it captured the face and drawn a rectangle around it. (Todo: insert pictures)

Once I had the basic face recognition I started trying to detect the eyes as well. The algorithm to try and keep high accuracy is shown below.

Initial planned algorithm

Start Up

1. Calibrate default face with correct orientation.
2. Obtain face, eyes, mouth and chin landmarks. (Might add ears later).
3. Calculate differences between landmarks.
4. Store difference and start update loop.

Update

1. Capture frame.
2. Locate features. \*
3. Calculate face orientation.
4. Calculate differences between found landmarks.
5. Compare difference to calibrated image. – indicates mouth movement.
6. Draw rectangles over found landmarks.
7. Draw lines connecting points.

Locate features

1. Gray-Scale the image.
2. Equalise the histogram to minimise lighting issues.
3. Apply sharpening technique (Maybe).
4. Use the cascade.DetectMultiScale() function.

Results.

I did not get to fully test this algorithm. I was having difficulty making a distinction between an open mouth and an eye. Also, due to discussions with the Client, changed my priorities and started looking at other libraries and looked to see if there was a better approach.

Edge detection

Another thought I had was using edge detection to find features. I was interested in contours in the area that a face was located.

I researched what types of edge detection are available in OpenCVSharp and the main one I found was the [canny edge](http://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/canny_detector/canny_detector.html) [detection](http://docs.opencv.org/trunk/da/d22/tutorial_py_canny.html).

The algorithm that this [detection](https://en.wikipedia.org/wiki/Canny_edge_detector) uses is:

1. Apply Gaussian filter to smooth the image to remove the noise
2. Find the intensity gradients of the image
3. Apply non-maximum suppression to get rid of spurious response to edge detection
4. Apply double threshold to determine potential edges
5. Track edge by hysteresis: Finalize the detection of edges by suppressing all the other edges that are weak and not connected to strong edges.

OpenCVSharp implementation

There are only a couple of steps required to convert an image to canny edge.

1. Get image (from file/webcam etc).
2. Convert to gray-scale.
3. Use

Issues

* Drawing contours from an entire canny edge frame can result in about 7000 lines. too many to render in real-time. (Max limit is about 350 on my laptop).  
  This can be improved by finding the face, then finding and drawing contours in that area.  
  Tried to draw in parallel, this did not provide much improvement.

HaarCascade Observations.

* Is not rotation invariant (rotate face too much and it loses focus).
* Can pick up false negatives, this can be mitigated by changing parameters, but then it makes finding faces more precise (need a more accurate match).
* The cascade classifiers provided are quite basic and general, therefore trying to track jaw movement is going to need additional steps/ maybe another approach.

For the rotation issue a possible solution is rotating the captured frame about 40 degrees clockwise and counter-clockwise, if there is a face staring at the camera It should recognise it. However, if faces found when the frame is rotated will need to be rotated back to the original position to maintain accuracy. A potential problem with this is you are running the cascade several times and it is an expensive operation. Depending on how much processing you are trying to do in a single frame then you are going to notice performance drops due to this.

[OpenCV](http://opencv.org/) also provides a way of training you own classifier. I will discuss this further on.

What is HaarCascade?

HaarCascade is a technique that was proposed by Paul Viola and Michael Jones in 2001. It is a machine learning approach, in which, a cascade function is trained on a series of positive and negative images (images either containing the object to be found or not). This is then used to detect object in images.

For a slightly better explanation I recommend visiting the page on OpenCV website that explains it. A link to it can be found [here](http://docs.opencv.org/trunk/d7/d8b/tutorial_py_face_detection.html). Alternatively, the paper in which it is discussed can be found [here](http://users.utcluj.ro/~tmarita/HCI/C7-8-extra/Face-detect/violaJones_CVPR2001.pdf).

Training a cascade. Todo: Update this section.

To train a cascade to detect an object, there needs to be a series of positive and negative images. Once it is separated.

Steps

* Download OpenCv
* Run CMake on the project and apply correct visual studio settings.
* Open Apps folder produced in the build folder to check if build was complete.

Links

<http://alereimondo.no-ip.org/OpenCV/34>

<https://github.com/qian256/OpenCVSharp3Unity>

<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2745713/>

<http://www.sbdmj.com/053/053-01.pdf>

<http://docs.opencv.org/2.4.13.2/doc/user_guide/ug_mat.html>

<http://stackoverflow.com/questions/7899108/opencv-get-pixel-channel-value-from-mat-image>

<http://docs.opencv.org/3.2.0/dc/d88/tutorial_traincascade.html>

<http://stackoverflow.com/questions/36259712/save-output-image-in-jpg-format-opencv>

<http://www.sciencedirect.com/science/article/pii/S188276160900009X>

<http://docs.opencv.org/2.4/modules/core/doc/drawing_functions.html>

<http://docs.opencv.org/2.4.13.2/doc/user_guide/ug_traincascade.html>

<https://en.wikipedia.org/wiki/Cascade_algorithm>

<http://opencv.org/>

<http://coding-robin.de/2013/07/22/train-your-own-opencv-haar-classifier.html>

<http://emovu.com/docs/html/getting_started.htm>