Creative Projects 2 Proposal

Project Overview

At a high-level, the app will attempt to algorithmically recreate an arbitrary input image (from the user) out of images that look similar to small subsections of the original image. This algorithm will be performed recursively on the image, printing or saving the result on each succession.

The project will be an image manipulation app that will analyse a whole image using openCv to identify parts of the image. These parts will then be compared to a bank of images to find one that closely resembles it. This new part of the image will then replace the existing part, thus creating a new image that resembles the original but using other images.

Using either an algorithm we have written ourselves using various computer vision techniques that may include analysing and comparing wavelet transforms, or using OpenCv’s histogram functions to compare images. Or by using the Google similar images API, we will aim to produce something that can produce interesting (not intentionally ‘accurate’) readings of images.

Audience

Whilst the project will be in the form of an app itself, it will aim to produce tangible pieces of ‘artwork’ based on the original image. This will, on one level, allude to the occasional erroneous nature of computer vision in unprecedented ways (comedic, dark?) depending on how the algorithm divides the image and what it deems a similar image to replace it with each time.

Conversely, It will also aim to exert a force for considering a ‘dumb’ piece of software’s autonomy and humanistic decision making along with its capacity for reasoning and human-esque visual analysis. Eliciting thoughts in the audience of how authentic the software’s ‘vision’ is and to what extent it is on-par with a human’s, the project will also encourage users to question to what extent the software is autonomous.

The series of prints will also present a manifestation of the concept of recursion through the process which will be conceptually similar to that of Chinese Whispers. Whilst the original image gradually becomes more and more diffuse after each successive time the algorithm is applied to the resulting image, the series of prints will provide this visual progress in the prints’ disparity.

The audience will not necessarily be people with explicitly expressed interest in computer vision and the project will aim to make an impression on those who are maybe unaware of the competence modern Computer Vision holds.

Background Research

Our background Creative Research[1] was focused on one new media group in particular - Bitnik. Recurring themes can be seen in the underlying concepts in their artwork; namely digital security and digital rights and surveillance.

Matthew Clark
James Oldfield
Our project takes some influence from *Darknet Shopper*[2] and the way in which raises questions about the rights of software and to what extent software is autonomous. It could be said that we are also following themes of Bitnik by creating physical manifestations of computational ideas, namely recursion. This idea is embodied in their piece, *H3333333K*[3], where a physical object takes on influence from a concept in the computer-world.

Another influence for this project, which was mentioned in our Creative Research, was the *Pirate Cinema*[4] by the way the creator was able to create new content from already existing of content. The way this has been reflected in our proposal is that we are creating a new form of an image by using parts of other images.

Following the idea of *Pirate Cinema*, Bitnik created another piece of work that manipulates existing content to create new work. *Download Finished*[12] was ‘an online resource’ that allowed users to upload and share video files. These files were then distorted and mutated when another user downloaded them. The mutation of a file, in our case an image, is the inspiration we have taken from this piece. Similarly, *H3333333K*, is does the same by mutating a file for their desired outcome.

**Methods and techniques**

Using libraries such as ofxCV will provide us with an interface to OpenCV, which will be a staple tool in helping us to decipher and locate objects in arbitrary imagery. ofxCV’s contouring method provides a nice interface to a way of segmenting arbitrary images and finding interesting objects inside them. This will undoubtedly prove useful when attempting to dissect interesting parts of images in the first step of the algorithm.

Kyle Mcdonald’s ofxCv[8] library simply sits as a wrapper around lots of the powerful features of openCV and provides as interface to some complex computer vision. We’ve identified Image Contouring to be a particularly good means to image segmentation in this library, as image contouring helps identify parts of the image which have the same colour intensity. We’ve also identified that image thresholding is a great first step in reducing pixels in an image to binary data, which can be much more useful in identifying interesting parts of image as adjacent pixels are converted to either black or white, showing a clear neighbour relationship.

We have identified two approaches to comparing images. The first is to write our own algorithm by analysing wavelet transforms[11] of the different segments of images. Achieving this well will allow us to compare two images easily because it will be possible to quantify similarities between images so a match can be found. In addition, we could use a technique could be to compare images using histograms which OpenCv has several functions that will be useful for this project[10]. Failing this approach, the Google similar images API exposes an endpoint with which we can send up sub sections of images to and have a similar image returned.

**Project Work Plan**

There’s quite a nice logical split for the workload behind our app. James will be working primarily on implementing logic to segment the images into sections and Matt will be working on comparing the images. Obviously it holds that if one section were to be finished earlier, we would both help in finishing the other part.

Matthew Clark
James Oldfield
**Week 1:** Kick off work finding the most appropriate ways to detect features and compare features with prototypes.

**Week 2:** Finish off prototypes and start work on the two sections of the apps.

**Week 3:** Working on our separate sections of the app.

**Week 4:** If we haven’t been able to implement our own algorithms from scratch, at this point we’re going to switch to alternative pre-existing solutions to finding images, namely the Google Similar Images API. (Reading week - lots of time to switch if everything isn’t working)

**Week 5:** Either implement google images API or finish own comparison algorithm. Have two separate working ‘apps’ by this point. Think about merging.

**Week 6:** Merge the two algorithms into one coherent single app.

**Week 7:** By week 9, we’ll hopefully have the MVP-esque functionality of the app, and we can start to implement the recursion.

**Week 8:** Bug fixing

**Week 9:** Bug fixing

**Week 10:** Bug fixing, final touches

**Week 11 (21st/22nd April): Hand in**

**Contingencies**

One rational worry is that we might not be able to effectively implement our own image comparison feature from scratch. During week 4, if we still have not implemented this, we will switch to using the Google Images API as a source of comparing images. If this API is no good, we will have to look for alternative APIs.

If any of the API’s are no good then we will resort to creating something slightly different which we’ll deem as our ‘MVP’. This will be an app that will take images as inputs and create a mosaic-like image as an output. How it would function would be to take a regular small region (e.g. 20x20 pixel area) and take the average colour value. Once this has been mapped to the image, the program will then look at other images to check if their overall average colour value is similar to the small region. The image with the most similar average colour value would then be place in the regular small region. This is done for the whole image creating a mosaic piece of art that still follows the principles that we want to convey. Similar to an webapp called “Mosaically”[5]. This is purely a backup plan under the edge case that we can’t implement our own comparison algorithms and then also can’t find any usable APIs for this task.

If Kyle Mcdonald’s ofxCv[8] library proves to be troublesome or not as effective as we’re hoping for our tasks, we will firstly look into openFrameworks’ native computer vision addon: ofxOpenCv. Whilst this doesn’t provide as broad of a range of computer vision techniques, it may provide the functionality we need in its contouring finding interface.

If these libraries prove to not be what we need, one other option is to write a wrapper around some native openCV functions, like the Watershed algorithm, which proves to be a popular means to image segmentation. Failing to successfully identify any intelligent segmentation at all, we will simply segment based on colour pixel value or by taking square shaped segments.

Matthew Clark
James Oldfield
References

1. Creative Research:
2. Darknet Shopper by Bitnik
   a. https://wwwwwwwwwwwwwwwwwwwwwww.bitnik.org/r/
3. H3333333k by Bitnik
   a. https://wwwwwwwwwwwwwwwwwwwwwww.bitnik.org/h3333333k
4. Pirate Cinema
5. Mosaic Art
6. Technical Research:
   a. http://gitlab.doc.gold.ac.uk/joldf001/technicalResearch
7. OpenCv Library
8. ofxCv by Kyle Mcdonald
   a. https://github.com/kylemcdonald/ofxCv
9. Histogram Comparison OpenCv Tutorial
10. Google Similar Images API
    b. https://developers.google.com/image-search/?hl=en
11. Wavelet Transforms
12. Download Finished by Bitnik
    a. https://wwwwwwwwwwwwwwwwwwwwwww.bitnik.org/d/