University of Manchester School of Computer Science

Project Report 2017

Change Detection in Live Video

Author: Joseph Allen
supervised by
Tim Morris

November 3, 2021

Abstract

In Computer Science we are used to collaborating with other fields; it is a fundamental aspect of programming careers to apply technical skills with a real world problem to create a useful application or system. Art and Computer Science can be seen as opposites, art strives to create products with no use or application other than to convey or elicit emotion, whereas Computer Science historically has not concerned itself with emotion (until more recent additions to the field such as User Experience).

The outcome of this project is the creation of a product which allows artists to trivially perform basic change detection with the sole purpose of artistic play. This project will help introduce non-technical artists to programming, inspire a sense of curiosity to delve deeper, and hopefully transition them into mastery of a language.

I started out with a practical focus of creating an algorithm with high accuracy and implementing the SAKBOT architecture, but through interviews discovered that artists would rather have a digital playground over high accuracy change detection.

My goal here is to create a project the encourages Artists to be more like Computer Scientists, and vice-versa. The success of the project is defined by the joy the artists feel using the product and the encouragement they feel to continue with Digital Art.

Acknowledgements

Thanks to my family for carrying me, Georgina for holding my hand, and my friends for making me smile.

Thank you for Tim Morris for all your support and guidance throughout this project.

Contents

Chapter 1

Context

1.1 Introduction

Image Processing is applying algorithms to images to generate some output.

Change Detection is the process of finding the changes in a scene over any period of time, in this case we use live video as this leads to instant visual feedback for the viewer.

In the past I have had a handful of artists approach me and ask if I can create ambitious products in a short period of time. My goal here was to create something that artists could use in the research stages of their projects to introduce them to programming, inspire curiosity and lead to mastery.

1.1.1 The Current Problem

This Project is split into two separate problems. First we have the Image Processing problem of change detection in live video. Second we have the Software Engineering problem of creating a product for an artist.

In Image Processing, change detection is the process of splitting an image up into two separate masks:

1. The foreground mask, one consisting of objects that are considered to be a change of interest such as objects, shadows and ghosts.

2. The background mask, consisting of the unchanged or unimportant environment of the scene; this mask can be simply considered the NOT of the foreground mask.

The other problem lies in creating a usable product for my client, an artist. In this there are problems of over-ambitious goals and frequent requirement changes. While I started the project focusing on the accuracy of my change detection, over time I came to realise that artists do not care about the robustness of the change detection, they care more about the visuals and ease of use of the system with a very experimental way of thinking.

1.1.2 Proposed Solution

I propose the creation of a digital playground that lets artists explore change detection in the Processing[PRO] IDE, in a form that is easy to use and install for both technical and non-technical

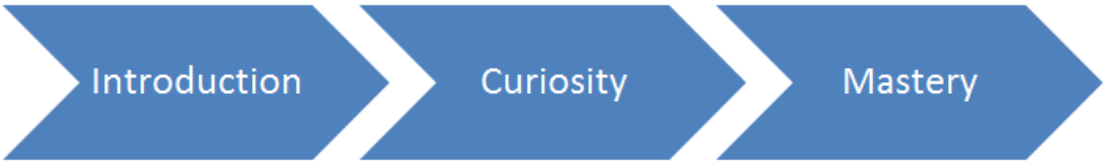

Figure 1.1: Introduction, Curiosity, Mastery process I am hoping to aide.

artists. I wish to create a product which will give non-technical artists an introduction to Processing, and at the same time other aspects of programming to guide them to curiosity and then mastery of my tool as well as the Processing IDE and Java Programming language. The plan was originally to implement the SAKBOT architecture[Cuc], however I later discovered that this advantageous architecture did not serve the needs of the users.

1.1.3 What is Processing?

Processing is an IDE, and by their own definition, "a flexible software sketchbook and a language for learning how to code within the context of the visual arts."[PRO]. Processing is open source and exists to give artists a simple introduction to using programming to create digital art. I took this project as a source of inspiration for my own, deciding to build my project in Processing and also follow their ethos of inspiring and teaching artists to be more comfortable with digital art and programming.

Daniel Shiffman [SHI] is a project lead at the Processing Foundation, and he creates some impressive video tutorials to help get artist started with programming and their use of Processing.u I am inspired to do the same and create video tutorials on how to use my project to help overcome the difficulties associated with Processing setup and setting up the project itself.

1.1.4 How do we currently do things?

First I approached how artists currently achieve art, in general they said their "processes are sloppy, unorthodox and never, ever written down and remembered properly" but "everything else is pretty much trial and error". This trial and error playful mentality can take many forms of exploration from taking photographs to painting. This is the functionality that the project will aide.

Currently, a common method for artists to explore programming is the use of an IDE called Processing which will be explored in more depth later in this chapter.

Collaboration is a commonly used technique to merge two different creative fields together to create something relevant to both fields. This collaboration can occur between two artists but also between two completely unrelated fields to create something meaningful.

An example I discovered while researching with my client was from a show called Sk-interfaces [ClZ]. We met the scientist who created the living skin coat and the piece itself provoked controversial discussions about whether or not the coat itself was "alive". When the exhibit came to its end the artist felt too attached to the project and believed in turning off the coat, she was killing a living thing. It is interesting to reflect on this dichotomy between an artistic and emotional approach in contrast to the cold and functional approach of the scientist.

1.1.5 Aims

The goals of the project are to help artists introduce themselves to code, inspire a curiosity to do more, and provide a tool that leads to mastery in a similar way to Processing[PRO] does itself.

1.1.6 Methodology

I initially started out this project with a Cowboy Coding approach; each week I implemented a more advanced Image Processing technique to explore the wide scope of possibilities this project has to offer. The main focus of this early stage was to gain a more in depth understanding of the trade-offs between frame-rate and increased accuracy of the change detection itself.

The second half of the project pivoted to a more structured approach, I now knew what functionality was possible from my spike and I used the Kanban[KAN] approach. I found Kanban suitable since there is only one developer working on this project and this gave me much needed visualization of the tasks that needed to be completed to make the project a success. This period was one of focusing on Usability and the clients needs over what I, as a computer scientist, deemed important. I elicited the requirements necessary here through a semi-structured focus group and a structured questionnaire which I sent to relevant artists.

1.2 Apprehension

1.2.1 Conception

The thought of creating Digital Art came to me whilst working in my year in industry at a web development agency. I found that the work was intense, but there were long periods of inactivity. In these periods of inactivity I tried to find a more personal use of the web language CSS, normally used to style web pages. I focused heavily on using built in transitions to make interesting animations which I later applied to my job in client-work. I made a goal to try to create a CSS only art gallery and went on to give a talk at my job about CSS-only artwork. To understand more about the art world I talked to my friend Sarah, a visual artist.

The Client

Sarah is a third year Interactive Art student at the Manchester Metropolitan University. Sarah was looking at nature as a perfect designer, and she looked to programming to design artwork to remove the human touch from her work and thought writing algorithms would be an interesting way to do this, which is why she contacted me. For the artist it is more about finding something that can be experimented with rather than an instant decision for what the final product will look like. She spent time researching programming for Digital Art and introduced me to Processing [PRO]. I would occasionally help her with what she was working on by creating quick and short code demos, from here I realized that programming for an artist would be an interesting topic for my third year project. My main concern was that the over-ambitious and experimental work of an artist would clash with a long-term project that Computer Science anticipates. My goal then became to try to offer something generic for artists that would utilise Image Processing.

Processing as Inspiration

After my introduction to Processing I decided to try to find a simple list of challenges to hone my skills but I discovered that such a list did not exist. I decided instead to host my own challenges and create an opportunity for others where there was not one for me. I went to the Processing subreddit[RPR] often for help and decided that this group would be where I would start a weekly challenge to help the community. This was a direct inspiration for my project in that it made me more confident in my Processing ability, but also made me want to help others in the Digital Art community learn Processing and my project is a realization of this.

1.2.2 SAKBOT

Early on in the project I was introduced to the SAKBOT architecture [Cuc], which uses Image Processing to isolate MVOs(Moving Visual Objects) in a scene. It uses advanced techniques such as Optical Flow and Blob Detection in order to separately classify the following:

1. MVO: Moving Visual objects that we are interested in, for example if we were looking to detect cars in a scene then the MVO would be the car itself.

2. Ghost: A Ghost is a "change" in a scene which does not represent the location of an MVO. For example the reflection of the car in a nearby shop window would be considered a Ghost as although there is a clear change, it is not an MVO that we would be interested in.

3. MVO-shadows: A Moving Visual Object shadow is a shadow which is connected to a Moving visual object, a shadow can simply be identified as a change in Brightness without a change in color.

4. Ghost-shadows: A Ghost shadow is any shadow not connected to a moving visual object, over time shadows in a scene may move due to the sun changing position, we do not want to count these as a "change".

The spike of color normalization helped realize the MVO shadows requirement and is a fundamental part of my Algorithm explained later.

1.3 Conclusion

So we have a familiar dichotomy, a computer scientist with a desired long-term goal and an artist who can be over-ambitious and frequently request changes. The goal here is to, like the Processing Foundation, use my technical expertise to create something fun and engaging. In the next chapter we will explore this technical expertise and how we can create a product.

Chapter 2

Development

Now that we have some context I will next introduce some Image Processing techniques relevant to Change Detection. I will discuss my use of methodology. I will discuss how I elicit my requirements and how I implemented then in ways which would delight the user.

2.1 Change Detection Basics

In this section I will discuss the fundamentals of change detection and move up onto more complex techniques used to increase accuracy. Change Detection is the process of finding the changes in a scene over any period of time, in this case we will use live video as this leads to better interaction.

2.1.1 Thresholding

Thresholding is a the simplest method of segmenting any image into two parts. It involves selecting a value and assigning any pixel value below that value to one segment, and any pixel equal or above that value to the other segment.

One issue with thresholding is you need to choose a threshold value to compare each pixel with. One method for doing this is finding the average value of all pixels and choosing this, you can expect an even split in each segment.

In my project the user has the ability to change this thresholding value in the default change detection algorithm.

Figure 2.1: Image of a Doughnut.

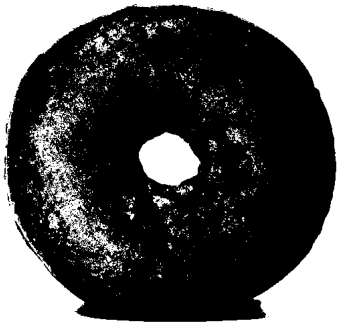

Figure 2.2: Thresholded Image of a Doughnut.

2.1.2 Euclidean Distance

Figure 2.3: greyscale color space.

Euclidean distance is the distance between two points in a space. The trivial example here is in greyscale color given points x and y the euclidean distance between the two points is $\sqrt{(x-y)^2}$. For example as shown in the image above we have all possible colors that can be stored in greyscale color, with 0 representing black and 255 representing white, A large change in this space is from black to white so performing $\sqrt{(0-255)^2}$ gives 255, the maximum euclidean distance in this space. This trivially extends as more color channels or different color spaces are added. We can use this to trivially decide what is considered a large change in a pixel. While I started with this method for RGB color I quickly moved away as this would consider shadows to be a change and changing the lighting in the room would also register as a change. This is what the standard OpenCV `diff()` function performs, we will discuss this later.

2.1.3 Color Normalization

Color normalization is a good way to ignore changes in brightness in RGB color space. If a shadow is cast over a white object, we don't want to consider shadows to be a change. Slight changes in scene lighting could register changes and we want to avoid this. Given L_R, L_G, L_B are respectively the red, green and blue color channels and we have some θ as our comparison value then $\frac{L_R+L_G}{L_R+L_G+L_B}$ this is called chromaticity which means color regardless of luminance, this gives

us a value to compare with θ which is no longer dependent on the brightness of the change. This means that pure white and pure black are not considered a change from one another, and is a disadvantage.

2.1.4 HSB Color Space

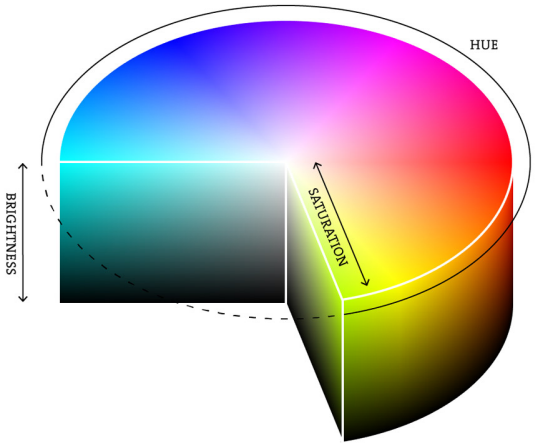

Figure 2.4: HSB Color Space.

The HSB color space is a different way of representing colors, I started using RGB, which has a separate channel for Red, Green and Blue colors. I moved to HSB (Hue,Saturation,Brightness) as then I do not have to worry about using color normalization as discussed in the previous section so this became the natural progression of my work.As shown in the above image the Hue is the orientation as an angle about the center, saturation is the distance from this point and brightness is the depth of the space. Ignoring changes in brightness when performing Euclidean Distance on any two points means we only look at the very top layer of this space, shadows are ignored successfully solving one of the problems of naive change detection.

2.1.5 Problems with Naive Change Detection

Change detection can be solved very trivially with the Euclidean Distance model, this generally gives quite a high mask accuracy, however it has a collection of problems which require more advanced techniques to overcome which are as follows:

Brightness Changes

One issue is brightness change, this can occur naturally in an outdoor scene. For example if it is a sunny day when the comparison image is taken, if a cloud then blocks out the sun for a few minutes, we will have a moment where every pixel is significantly darker. In euclidean distance this darkness could be enough to register as a change if the comparison value is low. In order to overcome this a method is needed to detect if there has been a brightness change over a color change.

Methods for dealing with this include the color normalization and HSB Color spaces I discussed in the previous section. One issue with these methods is that all greyscale colors are considered the same color so if you try to demonstrate an example of holding a solid black piece of paper in front of a white wall it is possible that no change will be registered. I have personally chosen to use the HSB color space because in my opinion it makes the code much more readable, for example it is clear when the brightness is being ignored in the algorithm and if one of my goals is to have the artists begin to understand the programming elements then readability is key.

Noise

Noise in an image is randomness which alters the image, this can come from the camera itself or the wiring connecting the camera to the computer. Noise can cause small but sharp changes in

individual pixels, since my algorithm looks at individual pixels this is a big problem as we don't want random pixels to register changes. The possible solutions for this involve either looking at areas larger than individual pixels, or a clever use of opening (erosion followed by dilation). While looking at areas larger than individual pixels may have been effective, I didn't want to move away from the individual pixel way of thinking that most people are familiar with. I went with several rounds of opening in the end as this is considered common practice in Image Processing and it gave the most reliable outputs after several experiments.

Background Update

Consider the example of a boat in the ocean , we calibrate the scene before the boat appears so we have some stage of the turbulent ocean captured, in the next frame the ocean will look different where the peaks of the waves have moved. In order to compensate for this we need some method of doing background updates. We have a `backgroundUpdate()` function in OpenCV for Processing library[OPE] which utilises the Mixture of Gaussians method of background subtraction, I have used this as one of the options artists can use but have not attempted a full implementation of the SAKBOT architecture[Cuc] because it is too intensive to run within Processing[PRO].

One important point here is that accurate background updates require a high framerate, if the algorithm is too intensive we won't find utility in updating the background as there will be too large a change to simply ignore. This is an issue which caused my optical flow spike to be ineffective.

Shadows

Shadows are essentially a small region of brightness change, and so our solutions to brightness change also solve the issue of shadows being picked up in change detection. In the SAKBOT architecture[Cuc] we treat shadows differently from the background subtraction which would deal with background updates, we instead look for local shadow regions which are nearby to MVOs and classify these specifically as shadows, this would be a great addition to the project as it would allow me to have a third mask of shadow locations which I am sure artists would love to play with, however attempts at identifying shadows became too computationally intense for Processing[PRO].

2.2 Change Detection Advanced

2.2.1 OpenCV

In this section I will discuss some external libraries I have used to save time on Algorithm implementation Greg Borensteins library OpenCV for Processing [OPE] and the more complex image processing techniques which it helped me to implement as well as a Blob Detection library [BLO].

Built in diff

the OpenCV for Processing Library comes with a simple `diff()` function which finds the absolute difference between each pixel in the compared object, in this case it compares every pixel in the Live image to the corresponding pixel in the comparison image, sometimes this will outperform my algorithm so I also included this in my product.

Built in background subtraction

The OpenCV for Processing Library comes with a background subtraction function `startBackgroundSubtraction()`. We have to set up the background subtraction before we start and then the must call `updateBackground()` with the drawing of each frame. This is also one option for algorithm choice in my product.

Erosion

In layman's terms erosion "removes the outer layer" from the objects in the image.

Erosion works by applying a structuring element over all pixels in an image. For explanation assume the structuring element is a 3 by 1 matrix of 1s, when applied we set the minimum of the pixel, the pixel to the left of it and the pixel to the right of it, shrinking the objects horizontally. In

normal erosion we will use a 3 by 3 matrix instead, This gives the effect of "shrinking" the objects in the image.

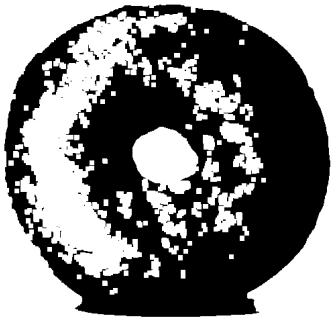

Figure 2.5: Eroded Thresholded Image of a Doughnut.

Dilation

In layman's terms dilation "adds an outer layer" to the objects in the image.

Dilation works by applying a structuring element over all pixels in an image. For explanation assume the structuring element is a 3 by 1 matrix of 1s, when applied we set the maximum of the pixel, the pixel to the left of it and the pixel to the right of it, shrinking the objects horizontally. In normal erosion we will use a 3 by 3 matrix instead, This gives the effect of "shrinking" the objects in the image.

The use of Erosion followed by Dilation is called Opening, Opening gives the effect of increases the size of holes within regions, hence the name, this can also be used to remove noise from an image which is how I use it in my project.

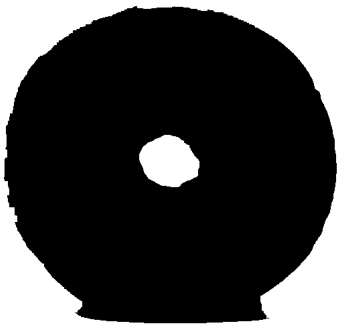

Figure 2.6: Opened Thresholded Image of a Doughnut

2.2.2 Optical Flow

Optical flow is the relative motion between two frames, it does this by storing the location of an object as a vector of it's movement between two frames. This was going to be useful to spot the motion taking place in my scene. In my project, I also found this to be a very brittle approach, since it relies upon a visible repeating texture to be able to compute optical flow, which in turn performs well only with a high frame rate.

2.2.3 Blob Detection

Blob Detection is about finding areas of similar properties such as color or brightness, the idea here is to find some important objects and hopefully I could use this to detect blobs that also contained changes, to remove noisy areas. This was temporarily part of the project as an early spike using the Blob Detection library[BLO].

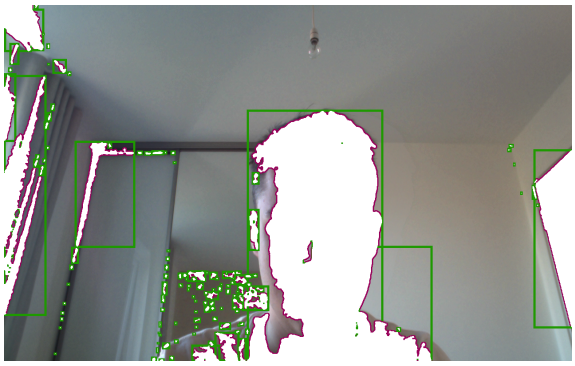

Figure 2.7: Blob Detection.

2.2.4 My Algorithm

My algorithm works by finding the Euclidean Distance of the Hue and Saturation in HSB color spaces. The HSB color space can be seen as a cone where brightness is the depth of the cone, the saturation is the distance from the center of the cone and the hue is the rotation about the center. Since we only look at Hue and Saturation the pixels location in space is only on the flat face of the cone, this means we can find the distance using the cosine rule. The algorithm is as follows:

```
For each pixel

    liveColor <- live[pixel]
    screenshotColor <- screenshot[pixel]

    liveHue = hue(liveColor)
    liveSaturation = saturation(liveColor)

    screenshotHue = hue(screenshotColor)
    screenshotSaturation = saturation(screenshotColor)

    hueDiff = abs(liveHue - screenshotHue)

    diffHueSat = cosineRule(liveSaturation, screenshotSaturation, hueDiff)

    if(diffHueSat > threshold)
        pixelChanged(true)
    else
        pixelChanged(false)
```

The performance of this algorithm could be greatly improved through a use of sub-sampling, since we are performing a comparison on every pixel in the entire image this leads to large complexity.

2.3 Building an Artistic Product

As a Computer Scientist I want to create a long-term project with a fixed goal as the client lacks understanding of the complexity and time requirements of the project with rapid changes coming in each week. The resolution I came to was trying to create a "Digital playground" to help artists explore and get through the "Trial and Error" stage of their research.

There were two main requirements going into this project, the personal requirement was needing some long-term product I could work towards. For my Client the requirement was that she had some sort of experimentation to help her develop her project.

2.3.1 Requirements Elicitation

In a simpler system requirements are much easier to discover and implement, however in this case we need some form of validation of the work. I decided to meet with various artists to figure out how art is achieved and how I could help them.

Types of Interview

There are three different types of interviews used.

- Structured Interview - The goal here is to make sure every person interviewed is given the exact same set of questions, the benefit here is that there are easily comparable answers and you can come up with meaningful statistics to validate your product.
- Unstructured Interview - The goal here is to find an interviewee who has a better grasp of the knowledge space than those making the product, to hope that they will reveal the true desires they have for the product. We do this by presenting the product and then trying to provoke discussion
- Semi-structured Interview - Starts out like a structured interview but still allows for some free discussion throughout the interview.

What I choose is a semi-structured interview, this allows me to lead the interviewees to discussions which I felt are important and then later allow them to go off in their own direction once they are familiar with the prototype. I used this method for both the Focus groups and the individual questionnaires to three different groups:

- Student artists - These are all students from third year at the MMU Art School. They focus more on academia and abide by more rigorous research processes and deadlines. Three of these such students were part of my focus group and my final client is one of these students.
- Professional Technical artists - These are artists who specifically focus on using programming to create their artwork and performances, this type of artists is generally very fluent with many types of programming language and are in general already familiar with Processing[PRO].
- Professional Non-Technical artists - These are artists who are professionally artists, they use their free time to create art and sell prints and work for events and exhibits.

These three types of artist make up the three persona's we will use to design and justify the system.

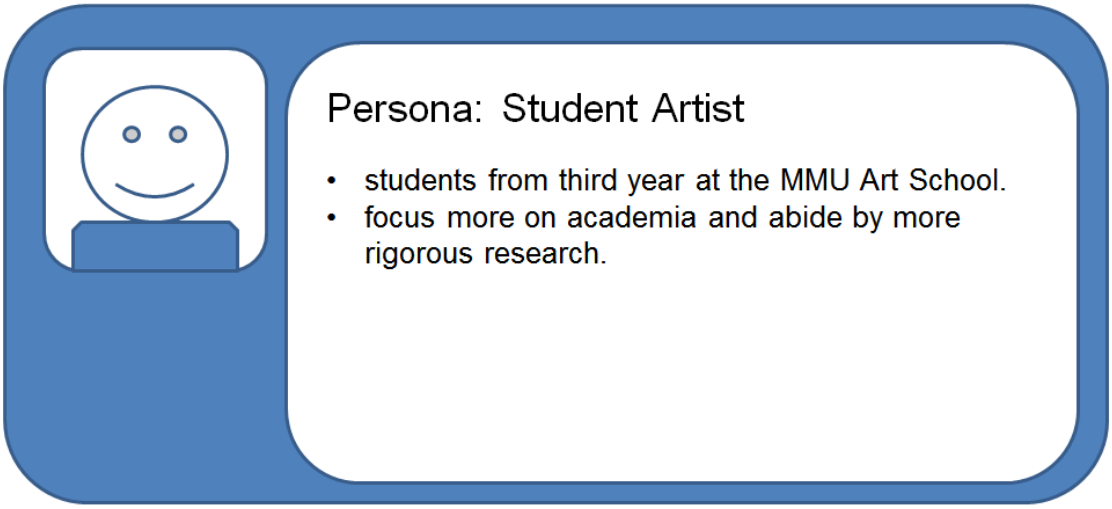

Figure 2.8: Student Artist persona

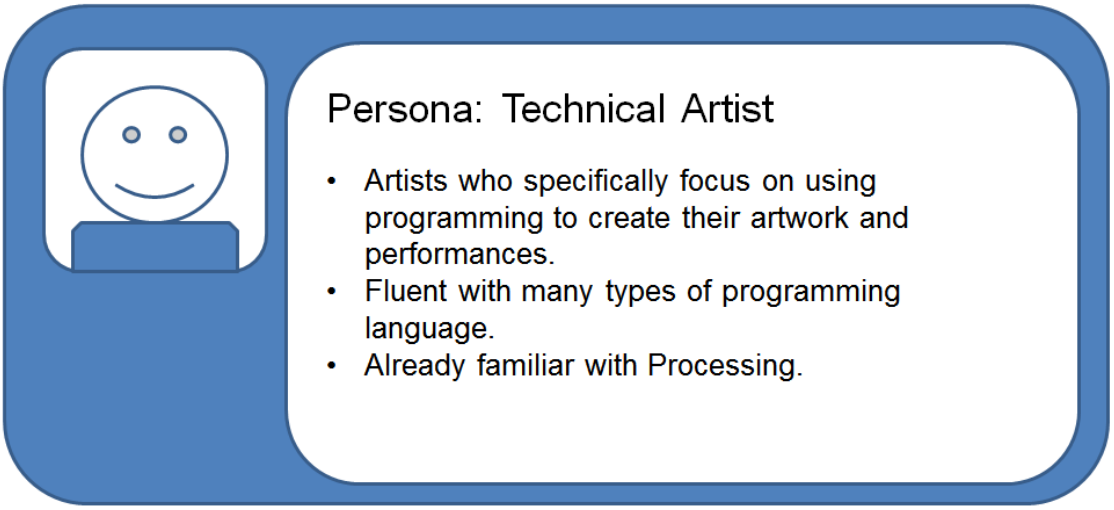

Figure 2.9: Technical Artist persona

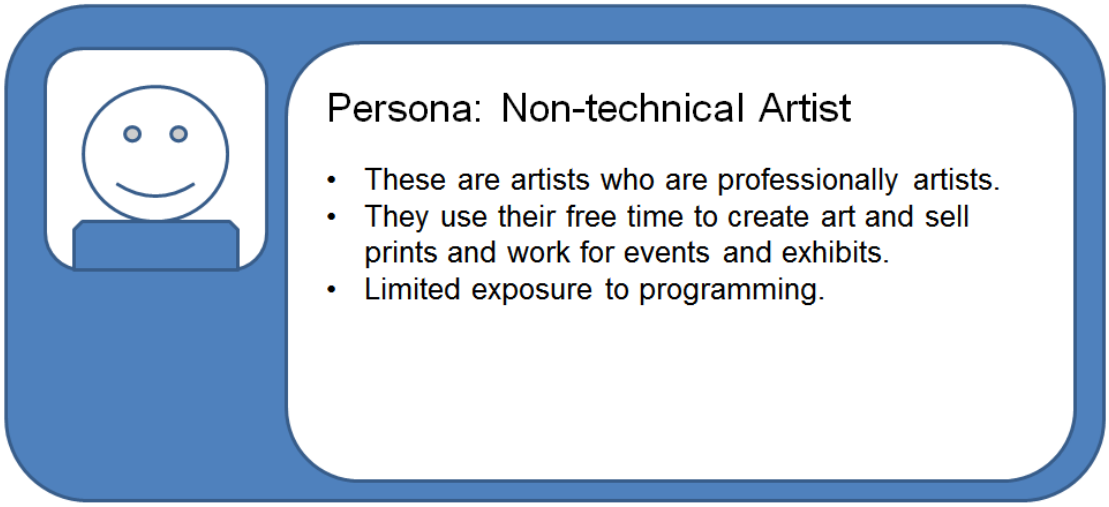

Figure 2.10: Non-Technical Artist persona.

Focus group

I asked 3 student artists to meet me in their art studio, I chose this location so that they would feel comfortable as there is an effect called Obtrusive Observation, where "The act of observation changes the observed in some way"[UX]. First I asked them the following questions to gain an understanding of how art is achieved:

- What is art to you?
- What is the process to create art?
- What tasks do you always do to achieve art?
- Are there any digital products you use?
- What features do you like or dislike?

I then showed the group a very basic version of my project which showed white pixels for the foreground mask and black for the background mask and no other features. I then asked for any initial impressions and feature requests.

Decision and Justification

Again I re-iterate the difficulty in creating a product for artists, I needed a way to create requirements which will give me a long-term project goal.

2.3.2 Codification

Codifying is the process of converting a real world problem or solution into a system with fixed rules so that it can be implemented in a program.

The first step here is try to figure out what artists do to achieve Art as a process. The focus group was very helpful for solving this, first I grouped all meaningful comments from the focus group and interviews into their relevant question groups and they are also color-coded by the different persona's answering them. From here we can notice any repeat comments and understand a generic solution to our question of "What is Art?" and "How is Art achieved?". I found in general that the artists start out looking for "Nothing to achieve in particular", the main focus is to elicit themes and explorations using "Trial and error" which lead me to the realization that a successful product for these artists needs to encourage this playful mentality whilst also allowing for curiosity to lead to mastery of technical skills in an attempt to transition the non-technical persona's to the technical ones.

2.3.3 Requirements

From this I created a collection of functional:

1. The User can have a solid color Background and Foreground.
2. The User can have a picture as a Background or Foreground.
3. The User can have a video as a Background or Foreground.
4. The User can have a coded example as a Background or Foreground.
5. The User can have the current camera as a Background or Foreground.
6. The User can easily decide what the Background mask is.
7. The User can easily decide what the Foreground mask is.
8. The System will make sure the User only has one active Background or Foreground.
9. The User can adjust the tolerance of the change detection algorithm.
10. The System should allow the separate masks to be saved.

11. The System should allow the user to generate movies.
12. The System should have a high frame rate.
13. The System should be responsive.
14. The System should perform “good” change detection.
15. The System should be easy for a user to download and understand.

I also created a collection of non-functional requirements, they are not so much requirements as they are goals to strive for.:

1. The System should have a high frame rate.
2. The System should be responsive.
3. The System should perform “good” change detection.
4. The System should be easy for a user to download and understand.

2.4 Functionality

This section will focus specifically on the functional requirements of the project and how they are achieved. A breakdown of these features is shown at https://youtu.be/H9fSh-QFq9U.

2.4.1 Color

The functionality here is allowing the User to select the Foreground and Background mask colors, while there is a function within G4P[G4P] that uses the built in Color picker, I chose to intentionally leave this as something which must be changed in the code, this is because it is a simple change to lead the artists using the project to look into the code. I have created a video tutorial to help make this easier.

Figure 2.11: Color segments.

2.4.2 Image

The functionality here is allowing the User to select the Foreground and Background mask Images, these images will be resized to fit the current monitor. This means that the artists will need to choose images of the correct resolution. However I believe it will be more common to have a user try to use a small image and so rather than have the image not be big enough I force this re-sizing. Since the code is open-source the artist can turn this off themselves if they wish.

Some examples of use here is that the Artist can set the Foreground and background to be an annotated and unannotated diagram to allow the people in the scene to reveal aspects of the underlying image.

Figure 2.12: Image segments.

2.4.3 Video

The functionality here is allowing the User to select the Foreground and Background mask videos, these videos are also resized to fit the current monitor. there is an functionality here as the audio is also playable so the users also need to be able to mute and unmute the videos.

Figure 2.13: Video segments.

2.4.4 Code

The functionality here is allowing the User to select the Foreground and Background mask coded examples. Processing [PRO] works by having a `setup()` function which runs once at the start of the program followed by a `draw()` loop which runs on every individual frames generation. In order to have coded examples we replicate this behavior, encouraging artists to learn the fundamentals of Processing before proceeding.

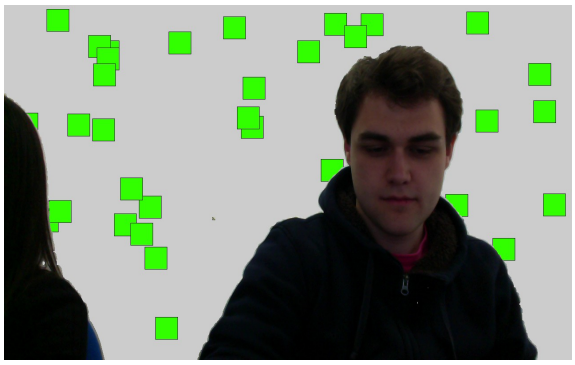

Figure 2.14: Code segments.

2.4.5 Camera

The functionality here is allowing the User to select the Foreground and Background mask as the live camera.

2.4.6 User Interface

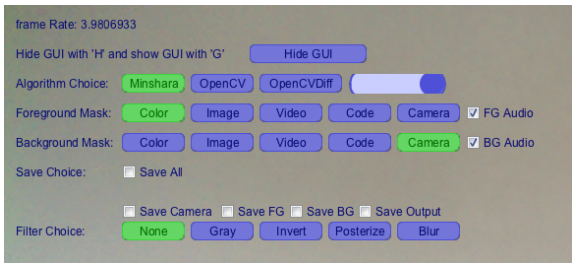

Figure 2.15: User Interface.

The User Interface is designed to let the artist quickly take control of the layers, while any color, image, video or code change must happen before each run, the user can control which layer is currently being shown, and can change it at any moment. The User also need to be able to control the save options at any moment.

2.4.7 Saving and Movie making

The User has the following Save options:

- Save All - Sets all of the other save options to the same as this one. Allows the user to save all possible layers at once should they need them for their comparison.
- Save Camera - Saves the entire camera input per frame.
- Save Foreground - Saves only the Foreground layer, saves black where there is no Foreground.
- Save Background - Saves only the Background layer, saves black where there is no Background.

- Save Output - Saves the addition of both the Foreground and Background.

the Processing IDE[PRO] has a built in movie maker which takes a collection of saved frames, a frame rate and an audio file. This saves me from having to develop my own movie maker and is one of the benefits of using Processing.

2.5 Usability

Usability is about easing the use of the product, this section focuses on my User Experience(UX) approach to HCI and the distribution of the product itself.

2.5.1 User Experience

User Experience is a more recent iteration of the ideas surrounding HCI, where there is not yet a consistent definition across academia and industry the field itself is an amalgamation of HCI, psychology and many other fields. My personal definition is that User Experience is about pre-empting user expectations and aiding the user in learning new behaviors. A product with good User Experience is a pleasure to use.

I will quickly work through some User Experience design here by answering the question "How does the user know they have captured their calibration image?". In order for the system to perform change detection it needs an image to compare against, we need a way for the user to capture the scene and a way to give feedback to let the user know that the screenshot was taken.

In standard HCI examples, you would expect something like an alert, a text message or a console message to let the user know this screenshot has successfully been taken. I believe that none of these are appropriate solutions, an alert is invasive and have negative associations. A text message may appear but may be difficult to notice. A console message is even less likely to be noticed, especially if the program is running full screen.

So we have some issues, we need something noticeable, non-invasive and something that is in alignment with the users current expectation. I asked myself "how would I capture an image outside of my system?", the trivial answer is I would take out my phone or camera, and take a photo triggering a flash. Use of metaphor is a common fundamental method of creating familiarity within a system. My simple solution is that when the user attempts to capture the screen the screen flashes white, just like a camera giving a satisfying method of feedback using behaviors the user already knows from the real world.

2.5.2 Availability

The Project is made available on a public Github repository[GIT], I chose this because one of the aims of the project is to help artists become more like programmers so an early exposure to Git is useful. The project is Open Source and Github has an excellent interface for dealing with Open Source pull requests again making Github a good choice for distribution. A downside is that there are easier methods of distribution so I am hoping that it is trivial enough to download a ZIP file from Github itself, I tried to get around this by making the ReadMe file as obvious as possible. I could have written the project in p5.js which allows the project to run in browser, however I wasn't sure Javescript was the right choice for an Image Processing project and this distribution would not encourage the artists to look into the code themselves.

2.5.3 Learning

In order to encourage the curiosity and mastery stages of the project I attempt to remove all difficulties of setting up the project and get the source code into the hands of the artists as simply as possible. I then focused on creating good documentation and video tutorials to assist in setup.

README

A README file is a simple text file which gives basic information about the contents of the project, GitHub supports markdown making this a little more interesting to read. In the README I give the basic instructions to install and set up the project and dependencies. The benefits of

the README is that it is a small and trivial addition to the project which may massively aid understanding, however there is also the downside that README files are generally ignored.

Video Tutorials

The purpose of these videos is to relieve the pain points of the non-technical persona of getting the project, these are the following:

- Downloading Processing.
- Downloading the Project from GitHub.
- Downloading the additional Libraries.

In order to make sure it is as trivial as possible for the users to set up the project I have created the four videos as listed:

- How to Download Processing? https://youtu.be/VQdCIbRJZGU
- How to Download the Project? https://youtu.be/5WOhYDDejkw
- How to Set Up the Project? https://youtu.be/qB2NxJeCd7w
- How to Change the Masks? https://youtu.be/H9fSh-QFq9U

2.5.4 Conclusion

We have now discussed the theory that surrounds change detection and have explored how a project can be completed despite ambitious not frequent change requests. I am confident that this project is in alignment with an artists' methods to achieve art and followed this with a second round of interviews which will be explored in the next chapter.

Chapter 3

Evaluation

The two main aims of the project were to create a product that would help artists be more like computer scientists, and vice-versa. The success is also defined by the joy felt by the artists using the project, as well as a sense of curiosity instilled which will hopefully lead to mastery.

I am confident in saying that I have met these aims as I have approached a project to create joy and fun over accuracy. The project was described as "like something I saw at an Art exhibition in The Lowry" in reference to the digital art exhibition Right Here Right Now, an event hosted in a local art gallery with other interviewees laughing with joy and asking if they can go outside and show other artist they know. As of the time of writing 45 have viewed the code and the videos have 322 total views.

3.1 Expectations

- I expected to focus on accuracy, spiking out blob detection and optical flow.
- I expected my clients to be overly ambitious and not understand the complexity of the project and request additional features hugely outside the scope.
- I expected the Project to perform at a much higher frame-rate.

3.2 Evolution of the project

The main change experienced over the course of the project was the effect of collaboration between two different fields, I found myself acting more like an artist focusing more on the creative and "fluffy" User Experience approaches to development over a more standard waterfall approach. In the meantime I found that the client became more aware of things that were trivial to implement in a short amount of time and explored programming in her own time.

A big change was my approach to the project; for the first 8 weeks I focused intensely on creating the most accurate change detection I could and attempted to implement the SAKBOT architecture[Cuc]. It was only when the project had bugs and I showed them to my client because they were interesting did I realized that the client would be more delighted with a strange bug than with a more accurate mask. This meant that moving forward my goal should not focus so heavily on accuracy and more on speed and flexibility in use, I had imposed my own requirements onto the project which my client never cared about in the first place. I resolved this by running a focus group and semi-structured interviews with a variety of artists and found that those artists focused on "Trial and Error" and are looking for "nothing to achieve in particular", this validated these thoughts and pivoted the project whilst also indirectly making me realize that I should think like an artist and not a computer scientist, or focus on client needs and not my own.

3.3 Testing

Creating tests for this kind of project would have been a large undertaking. I had a brief look at ViPER[VIP] but decided that painting frames to compare against was not worth the time, as the project evolved this became relevant as the goal shifted from accuracy to delight and exploration

which is not something so trivial to test. I attempted validation here through focus groups, interviews and user testing instead.

For the final round of interviews I performed basic user testing with three different artists from the Manchester Metropolitan University. I again performed these interviews in the art department so that the interviewees would feel comfortable. The first interviewee compared my project to a Digital Art piece by Daniel Rozen which was featured in the library [ROZ] and described my project as being "Not as mature as that stuff" which I take as a fair criticism, as the purpose of the project is to enable artists to make digital art. Another comment was about features beyond the basic segmentation such as having snow fall onto the foreground changes, this is clearly beyond the scope of my project and seen as a possible addition to the project.

My second interviewee was my client, Sarah. The goal of the project was to delight the artist and throughout her use of the product she was laughing and smiling at the various features. She commented that it "Could be good for Educational projects" which shows she has started thinking about use of the project beyond the research phase. Upon my explanation of how the coded foreground and background masks work she commented "I didn't know you could do two things at once" which shows an element of mastery, Sarah was learning new things about Processing through the use of the project. She also commented "when you first started doing it you had ideas straight away but you have to develop them" which shows that over time I have acted more like an artist, at the start of the project I would simply state uses for the project whereas now I put more artistic thought behind them. Sarah also suggested that this project was relevant to another artist she knew and asked if she could go get her, this shows that my client is already eager to share the project with other people. This feedback validates the entire project, Sarah has shown curiosity and mastery over the product and an eagerness to share the project.

My final interview was with an artist how focuses on creating visuals for live events, so my project which looks at live video should be a great fit for her. She immediately started asking questions like "Can you use external cameras?", "What do these Algorithms do?" which puts her in the curiosity phase of the project. She also suggested a new feature where you have a folder of videos and it iterates over those many videos so that it can be used with live playlists, this is a feature I did not uncover in my early stages of my requirements elicitation. In light of this feedback I would look into this feature given more development time. It is great to have an interviewee grasp the project so quickly and start suggesting basic features.

3.4 Conclusion

The project focus pivoted from accuracy to delight and exploration and HCI to UX, in my mind an apt metaphor for a computer scientist behaving like an artist. The next steps for the project are spreading awareness to both distribute to artists and improve the feature set through being open source.

The interviewees fit the three stages of the project, the first interviewee was newly introduced to Processing and was still overly ambitious like the non-technical persona. My client Sarah was at the beginnings of the mastery stage, she grasped the concepts and gained a greater understanding of Processing. The final interviewee was very relevant to the project and exhibited the curiosity stage, I didn't have to explain how much worked and she immediately started thinking about possible applications and trying to understand. These showed great validation for the project and shows that I have achieved my goals.

Chapter 4

Reflection

This chapter focuses on what changes happened over the course of the project. Over the course of the project I have learned many advanced image processing techniques, the Kanban methodology and a greater exposure to Processing.

4.1 General Reflection

I put in about 10-15 hours for this project and a large amount of my work was in small spikes which did not become a part of the main deliverable. There was some difficulty in balancing my project which the more intensive Mathematics coursework but I worked well ahead on these so that I would not be short on time when it was most needed.

The difficulty came from agreeing a final product with my client since there were no specific requirements or end-goals. I am happy with the complexity of the project and most advanced techniques were not useful for this specific project while they were still extensively explored. Through focus groups I validated that the complexity and accuracy were not important to the client.

The final product is the Change Detection GUI which allows the artists to control the layers used and the algorithm choice, with a focus on simplicity and how easy it is for artists to pick up and play with the project. In order to aide in the clients understanding I also delivered 4 tutorials for the project. While the project is itself finished, it is open source so myself and others can continue to contribute to it after the end of this academic project. There are already discussions with artists to use this project in exploitative creative workshops and hopefully exhibits.

4.2 What Changed?

The largest and most unexpected change over the course of the project was the focus shift from accuracy as a key metric to the goal of delight as a metric. Through User Experience requirements gathering I elicited that artists did not care for the accuracy of the change detection as long as it makes a decent attempt. Artists are not impressed when you spend a week improving your accuracy and find more delight in the glitches or the mistakes which made me realize I needed to run some sort of validation. This change came from my early goals of making an achievable long-term project and I realized a need to focus on the client needs over my own.

By approaching the project through focus groups and interviews I was able to elicit requirements I normally would not have considered important to the project. These interviews gave me validation and motivation over the course of the project and were of great benefit as the project proceeded as I could justify decisions as I made them.

I started the project with a belief I had to implement everything myself. I discovered a completed change detection example in the OpenCV for Processing [OPE] library, and was wondering if I should focus on this implementation. In the end I decided to include the OpenCV methods as well as my own algorithm as more options encourages that play mentality artists are used to.

4.3 Knowledge gained

Before this project my main knowledge of Image Processing as a field was a use of convolution matrices and edge detection from first year. Over the year I have gained exposure to many more Image Processing techniques and terminology including:

- Erosion and Dilation
- Opening and Closing
- Blob Detection
- Optical Flow
- Background Updating

In an early discussion with my supervisor I questioned whether or not we were allowed to use external libraries as I had discovered OpenCV[OPE] shipped with a change detection example. Before this project I was generally hesitant to use external libraries but after exposure to OpenCV[OPE] for Processing [PRO], GUI for Processing [G4P] and Blob Detection [BLO] I realized the time-saving benefits of using these external libraries over writing my own implementations.

Before this project I had only ever used waterfall and scrum to manage projects. Waterfall was inappropriate for this project since it required all of the analysis to be done before we started, and also due to the changing requirements from the client it is clear that an agile methodology is the correct choice. I chose Kanban since it allowed me to learn a new methodology and it focuses on priorities which is great for a single person project.

Over a year ago when I was first exposed to Processing I was eager to find a list of challenges to improve my skills, but sadly there were none. I took it upon myself to create some challenges for the community. I had never used Processing for image processing before so this was a great exposure to aspects of Processing I had never used before.

4.4 Conclusion

So to reflect the aim stated at the start of this report, "The goals of the project are to help artists introduce themselves to code, inspire a curiosity to do more, and provide a tool that leads to mastery in a similar way to Processing[PRO] does itself.". The three interviews validate that the introduction, curiosity and mastery cases for the project have been achieved.

Another goal of the project was for the collaboration between myself and Sarah to cause an osmosis of skills making me think more like an artist and helping Sarah think more like a computer scientist and this has also been achieved.

The next steps for the project are to continue sharing it on relevant social media and by word of mouth and if any artists pick it up I hope they will get in touch so I can assist them if they have any feature ideas. As of the time of writing 45 have viewed the code and the videos have 322 total views.

Bibliography

[BLO] Blob detection library. http://www.v3ga.net/processing/BlobDetection/. Accessed: 07-04-2017.

[ClZ] Oron Cattes and Ionat Zurr. Victimless leather. http://www.fact.co.uk/projects/sk-interfaces/the-tissue-culture-art-project-oron-cattes-and-lonat-zurr-victimless-leather aspx. Accessed: 20-02-2017.

[Cuc] Neri Picardi Prati Cucchiara, Grana. The sakbot system for moving object detection and tracking. http://imagelab.ing.unimore.it/imagelab/pubblicazioni/Avbs2001CucchiaraGranaNeriPiccardiPrati.pdf. Accessed: 20-02-2017.

[G4P] Peter lager, gui 4 processing. http://www.lagers.org.uk/g4p/. Accessed: 22-03-2017.

[GIT] Project git repo. https://github.com/joezcool02/MinsharaMask. Accessed: 27-03-2017.

[KAN] Kanban methodology. https://www.atlassian.com/agile/kanban. Accessed: 20-02-2017.

[OPE] Opencv for processing library. https://github.com/atduskgreg/opencv-processing. Accessed: 27-02-2017.

[PRO] Processing. https://processing.org/. Accessed: 18-02-2017.

[ROZ] Daniel rozen, darwinian straw mirror. http://www.bitforms.com/rozin/darwinian-straw-mirror. Accessed: 08-04-2017.

[RPR] Processing subreddit. https://www.reddit.com/r/processing/. Accessed: 24-02-2017.

[SHI] Daniel shiffman. http://shiffman.net/. Accessed: 20-02-2017.

[UX] Simon harper,ux from 30,000ft. https://leanpub.com/UX/read. Accessed: 22-03-2017.

[VIP] Viper: The video performance evaluation resource. http://viper-toolkit.sourceforge.net/. Accessed: 04-04-2017.