Robust Recognition and Identification of Paintings using Computer Vision Techniques

Author : Conor Broderick

Supervisor : Dr. Kenneth Dawson-Howe

M.A.I Computer Engineering

Submitted to the University of Dublin, Trinity College, May 2016

Declaration

I, Conor Broderick, declare that the following dissertation, except where otherwise stated, is entirely my own work; that is has not previously been submitted as an exercise for a degree, either in Trinity College Dublin, or in any other University; and that the library may lend or copy it or any part thereof on request.

Signature:

Date: *Wednesday 19th May 2016*

Summary

This project aims to develop a mobile Android application capable of recognising and identifying known paintings via computer vision techniques and image processing.
A user using the application would identify a painting of interest by pointing their smartphone's camera at a painting of interest in an art gallery or museum setting, snap a photograph of it and have the application relay information regarding the painting such as the painting's name, artist's name, era it was painted in etc. back to the user for them to read.
The resultant application that was developed shows that such an application is viable on a smartphone mobile device with no external processing on remote machines required.
In order to test the application, a public dataset of paintings from the Stanford Mobile Visual Search library is used as its dataset of known paintings.

The application itself is comprised of two main components to first find and extract what may be a painting in an image and then second to match this extracted painting to its correct match in a collection of reference paintings stored on the on board database containing the reference paintings and their relevant information. In order to find and extract what may be a painting in an image, a number of different image processing techniques are used. To detect what may be a painting in an image essentially a combination of smoothing, edge detection, morphological operations, and contour location, filtering, and extraction were used to detect the painting's frame and remove it from its background. In order to match the painting a combination of SIFT features and kNN (k nearest neighbours algorithm) were used to find unique features in the extracted paintings and match these to their respective counterparts in the reference paintings dataset. This report explores the theory behind these processes and how they work together in order to achieve the end of goal of matching and identifying paintings.

The results consist of the success rate of both the matching and extraction techniques employed in this application, and the time in which it took to match each painting to a reference in the dataset. As well as this, some test cases were observed for the extraction technique identifying both the effectiveness and limitations of the developed method.

Overall the performance of the application proved promising showing a 58% success rate or true positive rate in terms of correctly matching paintings to their references. The entire application is not finished but this project serves as a good point to continue from in order to implement a more complete application fit for real-world use. The final parts of the report detail some additional features that are to be implemented in order to improve the application on both a technical and aesthetic level in order to improve accuracy, performance, and ease of use for the user.

Acknowledgements

I would like to thank my project supervisor Dr. Kenneth Dawson-Howe for his feedback, guidance, and support throughout the entire duration of this project.

I would also like to thank my family and friends for proofreading this document and for supporting me throughout this academic year.

Abstract - Robust Recognition and Identification of Paintings using Computer Vision Techniques

Conor Broderick

This report details the research, theory, development, testing, and results of a mobile computer vision application capable of recognising and identifying a known set of paintings.

Art galleries and museums tend to rely on cumbersome paper based brochures, human tour guides, and/or audio tapes in order to deliver information about the artwork hanging on their walls to its patrons. An alternative approach to this problem is presented in this report in the form of a mobile application capable of recognising and identifying paintings whereby a user looking at paintings in an art gallery can obtain more information about them by taking out their smartphone and simply taking a picture of it. In this report, an investigation is carried out into the available literature surrounding this problem and a technique is developed in order to try and tackle it in a way without using any external computational machines for the required image processing.

The final results of the application showed a 58.24% success rate in correctly matching paintings extracted from images to their known reference counterparts hence demonstrating that computer vision could be applied in this case in order to serve as an alternative to traditional museum guides.

Introduction

Computer vision is a field that includes methods for acquiring, processing, analysing, and understanding images and, in general, high-dimensional data from the real world in order to produce numerical or symbolic information.[1] One of the main goals for computer vision has been to replicate the sophistication of the human eye in terms of perceiving and understanding visual information to interface with reasoning processes to elicit an appropriate response. In the case of this project I aim to solve a computer vision related problem using some of the techniques developed in trying to achieve this goal.

Aims

This project aims to develop an mobile Android application capable of recognising and identifying paintings in an art gallery or museum setting. A user of the application would be able to point their smartphone's camera at a painting of interest, take a photograph of it and then have the application recognise it and relay the relevant information back to the user about the targeted painting. The information returned to the user would include things like the artist's name, the year the painting was painted, the period the painting is from etc. This is achieved by extracting the painting of interest from the photographed image and comparing it to a set of known paintings to find its correct match using image processing and computer vision related algorithms and techniques.

The application would come preloaded with a database of known paintings for that particular gallery or museum and their relevant information.

Figure 1 below illustrates the application in use by a woman using the application to gain more information about a framed piece of artwork hanging on a wall in front of her by taking a photograph of it.

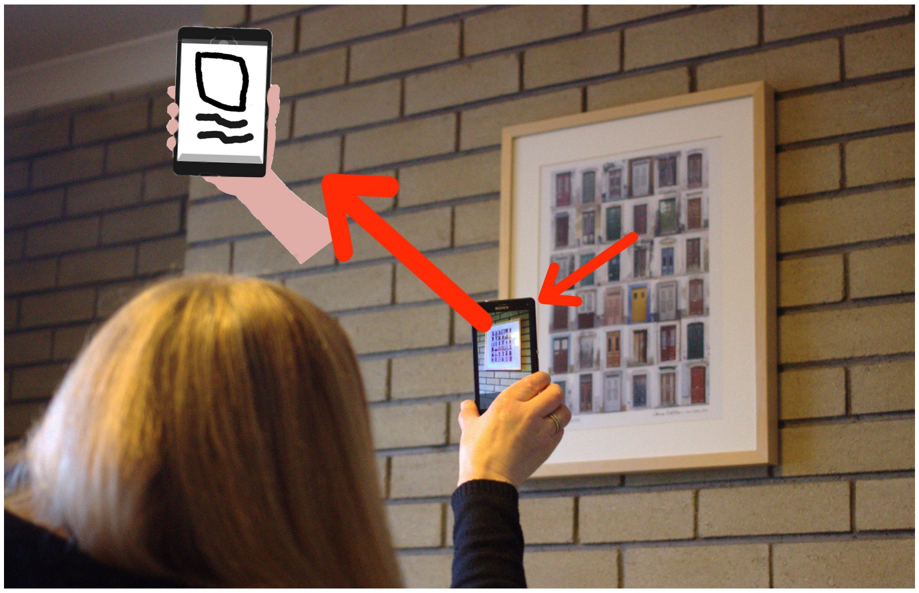


Figure 1 - Woman taking a photograph of a piece of artwork using the application to gain more information about it

Figure 2 below further illustrates how the app is used and what a typical result looks like for a recognised painting.

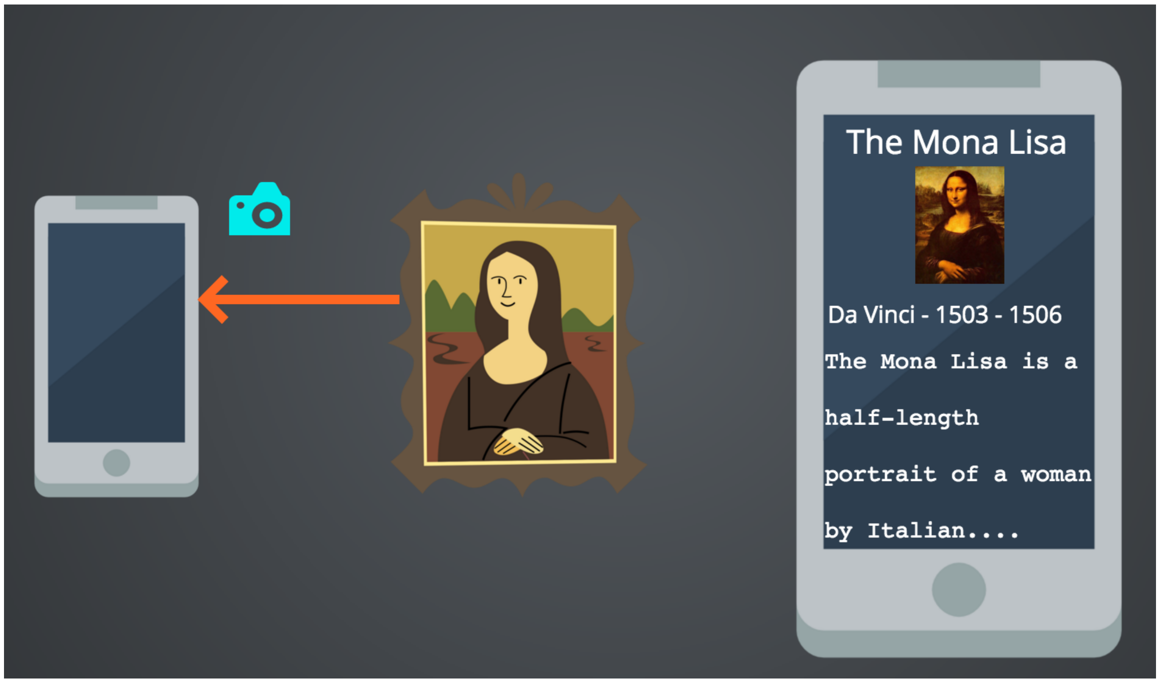


Figure 2 - Illustration of typical result produced by the application

Motivation

In today's modern world, smartphones are increasingly prevalent among the general population, becoming a standard possession of both the young and old alike. The ITU (International Telecommunication Union) reports that there are now more active mobile device subscriptions than there are people in the developed part of the world[2]. Due to their widespread availability and the companion-like attachment smartphone users have to their devices, there has been a lot of development of a large variety of applications to enhance the user's life and to simplify a lot of their day-to-day inconveniences that previously required separate real-world tools and items to solve.

Applications for scanning barcodes, retrieving real-time public transport information, and virtual wallets for keeping credit cards to cinema tickets are just some examples of applications that have been developed to achieve this. As well as this, smartphone cameras have become better in both software and hardware over the years which has opened the door to more sophisticated computer vision focused applications to become viable on the smartphone platform.

One such application could be that of an electronic guide to paintings in museums whereby users could point their smartphone device at a painting of interest in order to retrieve more information about the painting they are viewing rather than flipping through a cumbersome brochure provided to them by the museum they are in. In essence, this would result in turning the passive experience of walking around an art gallery or museum into an active one where the user is empowered to draw more information about the artworks surrounding them should they desire it. An application of this nature would also save money for the museums in printing out artwork guides for visitors as well as saving the paper that would be needed to print them on. Museums would also no longer be limited to a small placard on the wall beside the painting to deliver information about it to its visitors.

Imagine a tourist exploring Trinity College Dublin for the first time and wanting to find out more about the paintings and artwork installations around the campus. They could simply pull out their phone, download the app along with the latest information regarding

Trinity’s art exhibitions allowing them to discover more about the artwork and enrich their overall experience while walking around the grounds and visiting the various buildings and locations on campus. Such an application would also save the time and resources of the staff managing Trinity’s various art collections as they now all have a centralised location and dispensary for the information about the artwork rather than having to figure out what information is worth displaying along with particular pieces of artwork and paintings.

Ethical Considerations

In conducting any kind of research one must take pause to consider the ethical implications of the work they are embarking on. One must consider if what they’re doing will cause any harm to others, infringe on the anonymity or the privacy of others, and if any participants in the research have informed consent as to what they will be doing, what data is being collected on them, and what is being done with that data during and after the research has been completed. In the case of the ethical reporting of results from research, one must ensure that they are not introducing any bias to the test data or falsifying results in order to produce data favourable to their research. External material must also be correctly referenced in order not to plagiarise the work and efforts of others.

Test Data

One potential case for unethical practice on my part would be that of tampering with the test data I will be using to evaluate my application’s performance in order to produce favourable results for my research project.

My project uses a publicly available data set, the ‘Stanford Mobile Visual Search’[20] data set of museum paintings, in order to test the effectiveness of the application. In order to artificially improve my results, I could crop out the photographs of the paintings hanging on walls by manually removing the background in order to improve the effectiveness of my painting extraction technique which would in turn increase the number of successful matches the application returns.

This however would produce unrealistic results that would not be reflective of how well my application and the algorithms/techniques within are actually performing and may mislead and waste the time of others reading my work who are trying to achieve something similar and conduct related research.

Copyright

In the case of copyright, ethical issues may arise in the case of taking photographs of copyrighted paintings for use in the app as reference paintings. If I were to test out my application in a museum in Dublin, there's a good chance I may face some legal issues when it comes to taking both high quality reference photographs of the paintings as well as standard smartphone photos to test with.

The issue mainly lies in the fact that most museums do not hold the copyright to the paintings that they are displaying. According to copyright and fair use specialist lawyer Julie Aherns from the Centre for Internet and Society at Stanford University, she states that a photograph of an artwork could be considered a "derivative work," which is "potentially a violation of the copyright holder."[3]

This of course allows for the potential to be unethical and acquire or stealthily take the desired photographs for my project in order to acquire a dataset to test it with. In another somewhat similar example, researchers in the past have landed in hot water for breaking copyright laws in order to aid their research. In an article by NPR[4] it's revealed that many University libraries are ending their subscriptions to expensive journals which has led to their researches acquiring them through illegal channels. Breaking copyright law in this way is unethical and can lead to monetary loss for the copyright holder and therefore damage their business, research etc.

Summary

From the exploration of the various ethical issues above it is evident that some degree of care must be taken in regards to the handling of the test data and copyright of paintings in researching and developing the proposed project in this report. In order to ensure ethical standards are held I did not tamper or alter any of the test data taken from the Stanford mobile visual search dataset in order to skew my results, and this dataset was also used to avoid the potential to violate the copyrights of any artists whose paintings I may have had to take a high quality photograph of in order to test with.

Overview

The purpose of this section of the report is to provide an overview of the main components of the developed application, how they fit together, and how they overcome the challenges involved in achieving the end goal of identifying the paintings contained in the dataset.

The various components of each step here will be mentioned briefly. The theory and reasoning behind them will be explained in the methodology portion of this document.

The main sections of the application are:

- Capturing the image and extracting the painting from the background
- Matching the extracted painting against a set of known paintings stored in the onboard database.

Painting Extraction

In order to successfully match the painting it must be first isolated from its background in order to accurately detect key-points and compute their descriptors for it to match correctly with its reference. In order to do this a number of different methods were attempted to try and remove the background from the image. Methods such as Hough lines to detect the straight lines of the frame, corner detection to find the frame's corners, edge detection, and rectangular contour search were some of approaches attempted and tested for this.

The extraction method implemented consisted of a five step process. Starting with the initial captured image on the user's smartphone, a Gaussian blur is applied to remove some of the smaller edges while preserving the larger ones we're interested in. Following this the Canny edge detector is run over the image to detect the edges in the image, a dilation is then applied to connect any broken edges, the contours are detected and finally filtered to extract the largest one which will be the frame of the painting which we can then crop out from the background. If everything works correctly we should be left with the painting that was contained in our captured image with the entire background removed, thus isolating the painting for matching.

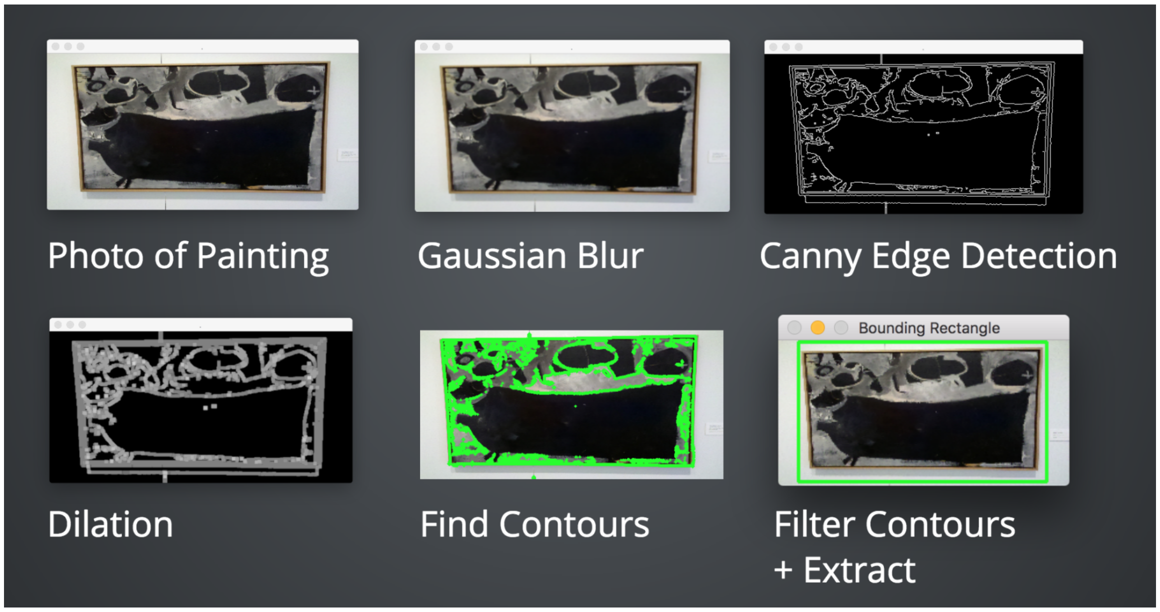


Figure 3 - The 5 step process of finding and isolating the painting from the background

Painting Matching

To match the painting to its correct reference a number of solutions were considered. Colour based, histogram matching, bag of words, and feature based techniques were all explored as to how to match the extracted painting to its reference and identify it correctly. One of these approaches made it into the final implementation of the application. Some of the others will be discussed in the latter parts of this document.

In order to match the extracted painting to its reference SIFT was used to detect local features in both the extracted image and the reference images of the paintings and compared using kNN (k-Nearest-Neighbour) to find the number of matching descriptors of key-points between the two images. A threshold was set for the number of matches to determine a match, otherwise the reference painting with the highest amount of matching descriptors of the key-points would be returned. Reference paintings that surpassed this threshold are considered to be ‘strong matches’ by the application.

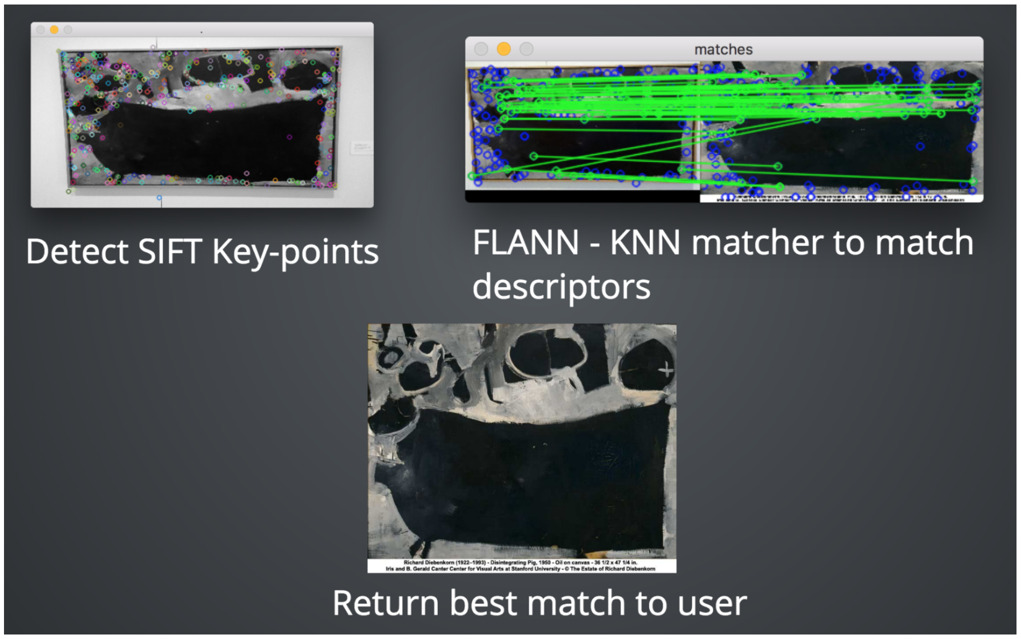

matches
Detect SIFT Key-points
FLANN - KNN matcher to match descriptors
Return best match to user

Figure 4 - Detecting SIFT key-points and descriptors + Matching with kNN

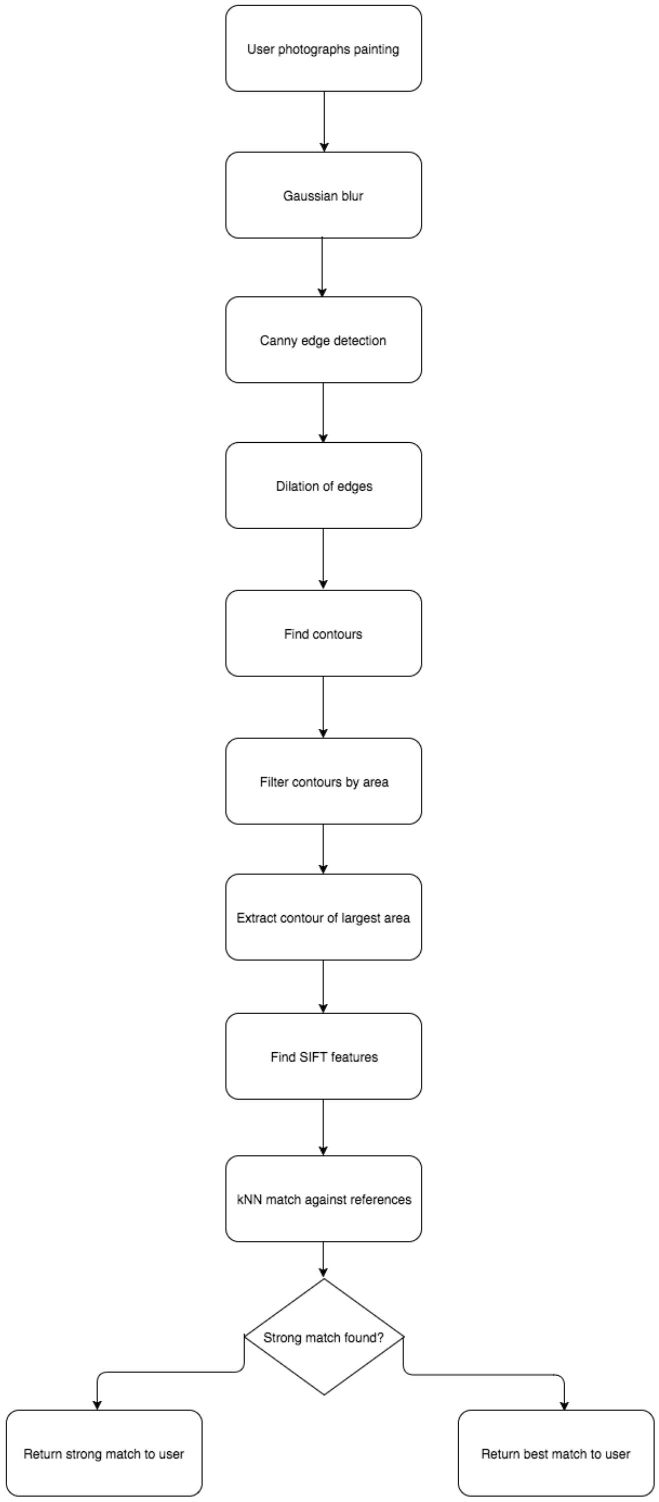

User photographs painting
Gaussian blur
Canny edge detection
Dilation of edges
Find contours
Filter contours by area
Extract contour of largest area
Find SIFT features
kNN match against references
Strong match found?
Return strong match to user
Return best match to user

Figure 5 - Flowchart of application's processes

Technology

In order to develop the application the OpenCV (open computer vision) library was used for implementations of the various computer vision related techniques and algorithms used in building this application. The external OpenCV library was also used for the use of patented algorithms such as David Lowe's SIFT. Initial testing and prototyping was conducted initially using Python before implementation in Java for the Android application. Developing and testing on the Android device itself is time consuming so the initial Python implementations served as a time saver for prototyping before implementing the final application. The paintings and some of their related information were stored and accessed via a SQLite database that came with the Android application itself.

Summary

In this section we have briefly seen and discussed how the application works from a high level overview of its different components. The painting is extracted from the background of the image using a mix of pre-processing techniques, edge detection, and contour finding and filtering. Then in order to match the painting to its reference, key-points are detected and descriptors computed for them using SIFT. Both the image and the references are then compared using kNN matching to match the descriptors and return the painting with the most matches.

Literature Review

The purpose of this section is to provide a review of the relevant literature I have studied for my thesis and how it helped to inform my design and implementation of the final application.
In order to avoid repeating myself too much I will say here that the following papers are all relevant to my project as they all generally have the same goal in mind of identifying paintings using computer vision related techniques and algorithms.
Each paper is essentially an exploration into how to use computer vision on a mobile device of some kind to identify and recognise paintings. The algorithms all break down into different components containing an extraction phase and a matching phase of some sort. Each of the papers and their findings have guided me towards how I tackled my own project in terms of implementation, known difficulties, and result interpretation.

Painting Recognition using Camera-phone Images [5]

This paper discusses a computer vision algorithm implemented to recognise paintings from a known dataset. Similar to my project, the algorithm was to be used as part of a mobile application in a museum or gallery setting in order to recognise paintings hanging on a wall and to provide audio commentary on the recognised painting.

In order to achieve this, their algorithm was broken down into two main parts. The first part was that of finding the painting in the image and transforming it so that it fit on a 50x50 pixel square.
The second part was recognising the now extracted painting by extracting its lower dimensional features using eigen-images and classifying it by using Euclidean nearest neighbour classifier.

The results of the algorithm they developed were quite positive as they yielded only 2 misclassified test images out of a total of 132 test images with an average running time of 0.5 seconds for recognising training images and an average running time for recognising test images of 0.5992 seconds. However, I would be skeptical as to how they selected their test data as in the paper they mention taking their own photos using a digital camera rather than a mobile phone camera to test their application.

While they briefly mention that they tried to take photos that would be on par with those taken with a typical smartphone camera, I would be uncertain of the validity of their strong test results given that they gave themselves plenty of opportunity to introduce a selection bias into their test data. Whether consciously or not, they may have taken and used photos that would be likely to produce favourable results for their project. A fairer assessment of their application would have been if they had used a dataset of publicly available photographs of paintings or collected their own photographs in a more unbiased fashion.

The paper concluded that the success and speed of their results may have come down to their own algorithms and that they may not be as robust for other image recognition problems. Their general findings were that the more complex time consuming algorithms proved to be more accurate and the less time consuming algorithms were less accurate but faster. Overall, they propose that a smartphone could be used to recognise a painting in a reasonable amount of time.

Painting Recognition From Wearable Cameras [5]

This paper describes a Google Glass application capable of identifying and recognising paintings for use in a museum or art gallery. The paper provides an experimental comparison of the accuracy and speed of the different feature detectors and descriptors on a realistic dataset of paintings from the Louvre. The paper identifies that emerging technology such as Google Glass provides the ample opportunity for computer vision and augmented reality applications to be developed for many day-to-day activities for the wearer and also that such applications due to the limitations of the hardware must be very fast and efficient in order to work effectively.

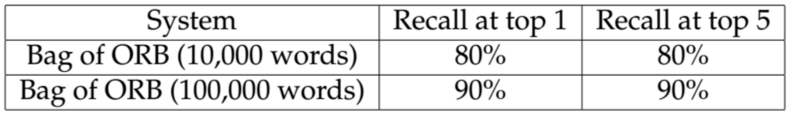
Their system is composed of two different parts. In the first part, for each of the different key-point detection and descriptor methods (SIFT, SURF, ORB etc.) they each compute the key-points and descriptors for all the images of the data set and store the result in an onboard database on the Google Glass device. The second part is when the user wearing the device and looks at a painting, the same algorithm is run over the input frame and compared against the images in the database. They rank the database images decreasingly according to the number of matches between the them and the input image.

What's notable about this paper and relevant to my own project is their discussion of how matching the descriptors of the query image to all the descriptors of the database image is not a scalable strategy. They identify that the complexity of the system and the time it will take to match a painting is linear with the number of paintings in the database. In order to resolve this they reference Josef Sivic and Andrew Zisserman's paper 'Video google: A text retrieval approach to object matching in videos.' which proposes using a 'visual bag of features' approach to the problem. The method involves constructing a vocabulary of features, then in representing each image by a histogram. For each of the images, their computed descriptors are converted into 'words' by taking the approximate nearest neighbour and the histogram counts the number of occurrences of each word. The histograms are then normalised and re-weighted so that the rarer words are more important than the common ones which means that rather than matching hundreds of descriptors we now only need to match one histogram which is significantly faster and makes the system a lot more scalable.

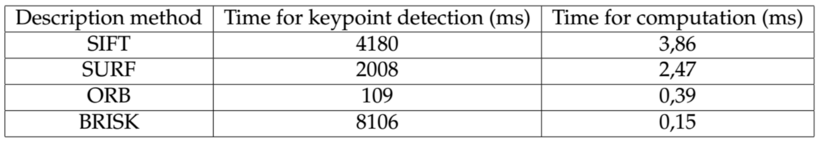
They tested 100 reference images of paintings and used 10 photographs they took themselves of each of the paintings in the museum. After obtaining their ranking of matched images against the input image, they measure recall at top 1 (the number of times that the correct reference painting from the database is at the top of the ranking list) and recall at top 5 (the number of times that the correct painting is in the top images of the ranking list).

System	Recall at top 1	Recall at top 5
SIFT	90%	90%
SURF	70%	90%
ORB (500 descriptors)	80%	80%
ORB (1000 descriptors)	100%	100%
BRISK	60 %	60%
SURF+BRISK	80%	100%

They did the same for their bag of words matching technique and finally compared the time for key-point detection of the different methods they used for matching.



System	Recall at top 1	Recall at top 5
Bag of ORB (10,000 words)	80%	80%
Bag of ORB (100,000 words)	90%	90%



Description method	Time for keypoint detection (ms)	Time for computation (ms)
SIFT	4180	3,86
SURF	2008	2,47
ORB	109	0,39
BRISK	8106	0,15

While I found the paper and its results to be fairly reliable I would be concerned that they took photographs of the original paintings that would be favourable to their application in terms of producing the desired results rather than a public set of photographs where no bias would be in place. The main takeaway of the paper for me would be that of the discussion of scalability and solutions to make the application more scalable which could also be implemented into my project for the gains in terms of scaling the application for use with larger data sets of known paintings.

Robust Painting Recognition & Registration for Mobile Augmented Reality [7]

In this paper an approach for identifying and recognising paintings for mobile augmented reality applications is discussed and developed. The aim of the approach is for it to be in real-time and the method itself is again, broken down into multiple stages. The first stage is what they've called 'the relevant painting region detector' which essentially finds and extracts the painting region from the given image. In the second stage, two local and global features are extracted from the relevant region to robustly match a painting from the database. Lastly, a RANSAC homography estimation method is used to overlay the additional content in an AR (augmented reality) framework.
They then perform experiments using a dataset of publicly available images to test their solution.

One of the key issues identified in this paper is that of the real-time nature of the application. Static vision issues like illumination, view, and scale changes are more severe when the camera is moving and the effects of blur, noise, and motion are introduced. While not as much as an issue for my project, I was still able to identify some of the key issues I would have to consider while developing my application from a static viewpoint from reading this paper and its discussion of these issues.

The diagram below outlines the architecture of the system they developed.
Three main modules process the current read in frame by the camera. The first module extracts the relevant painting region by removing the background of the painting and the painting's frame to keep only the portion of the image that contains the painting. It achieves this by using the Randomised Hough Transform[21] (RHT) method. Essentially what happens is after applying Canny edge detector, the RHT is used to to fit ellipses and rectangles around these edges. The painting is extracted by choosing the shape which has an area that is at least a certain times the size of the input image.

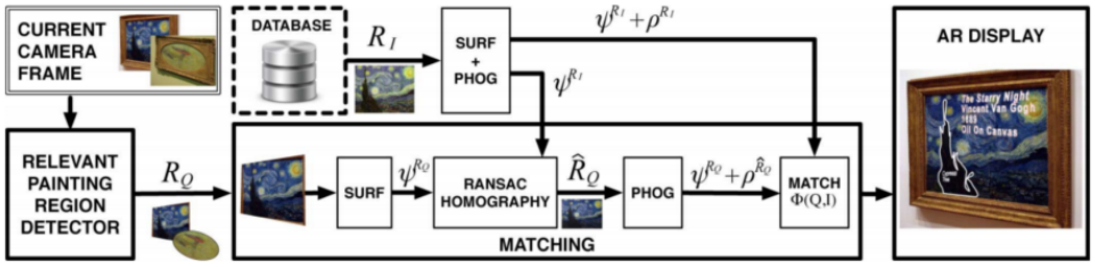


Figure 6 - Proposed architecture for system from Robust Painting Recognition and Registration for Mobile Augmented Reality *paper*

Following this, the second module computes the match between the current camera frame and a target in the database using SURF and PHOG. Finally, if the input frame produced a match with the candidate target, the homography transformation used to overlay the additional content to the current camera frame is computed by the third module.

The dataset used consists of 607 different publicly available photographs of 70 different Van Gogh paintings. The pictures are taken from different viewing angles and under different lighting conditions, some even come with light reflections and occlusions.
The algorithm has been tested on a standard PC with P4 CPU 2.0GHz, 1GB RAM, Windows XP and on a Tablet with ARMv7 processor 1GB RAM, Android 4.2.2.

In the paper's results, they first show the performance of their method without their relevant painting region (RPR) detector, and graph it against the rotation of the input image.
On average, a true positive rate of 49% is achieved for a false positive rate of 20%.

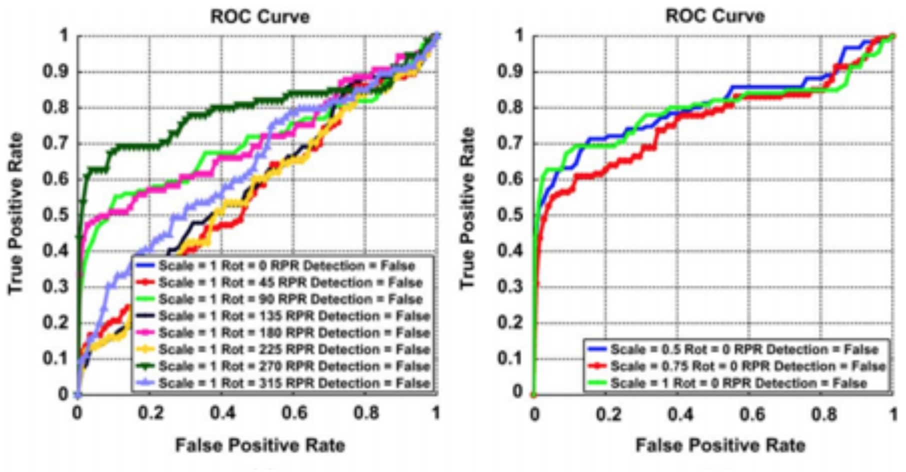


Figure 7 - False positive rate for RPR detector

For their second experiment, they graph the performance against image scale. In such scenario, an average true positive rate of 67% is achieved for a false positive rate of 20%.

They then perform the same experiments again only this time using the RPR before matching which results in a 71% true positive rate is reached for a false positive rate of 20% and for the second experiment a true positive rate of about 59% is reached for the same false positive rate of 20%.

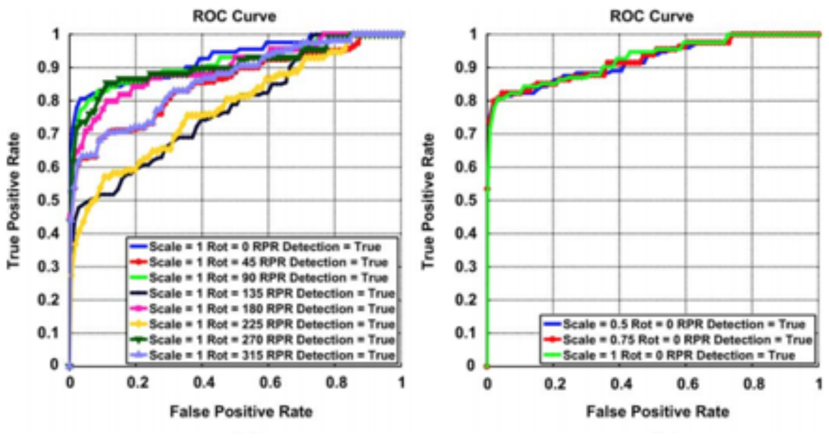


Figure 8 - False positive rate RPR before matching

In comparison to the original experiment without RPR, performance increases by about 33%.

The results clearly demonstrate the benefits of extracting the relevant painting region before matching as well as the effects of rotation and scale when it comes to accurately matching against a dataset. I believe the results of this paper to be sound judging by the public nature of the dataset used to conduct their experiments which was far less likely to introduce bias than that of the datasets used in some of the other papers in this literature review.

Their results and finding are especially relevant to my project as they explore some of the same issues I'd be taking on as well as running their tests on hardware similar to that of my own project, namely the mobile Android device.

Out of all the papers, their system's architecture would be closest to that of mine, especially in the painting extraction technique.

Summary

In this section of the report we looked at some of the relevant literature related to my project of other similar projects attempting to achieve a similar goal in that of recognising and identifying paintings. Looking at such material provides valuable knowledge which will likely inform how one might approach a new project using similar techniques in the same area. By studying the techniques used and the results they produced by the previously discussed papers I was able to make more informed choices as how to develop my application and how different techniques would perform in certain circumstances. This allowed me to avoid some of the more common pitfalls I may have otherwise fallen into had I have not read them which would have resulted in a significantly poorer performing application.

Methodology & Implementation

Introduction

This section of the report details the theory and reasoning behind the various parts of the application giving a description of how they work on a technical level and examples of the results they produce. This section essentially gives a detailed breakdown of the computer vision theory behind all of the working components of the application and why they were used.
To recap, the application is broken down into 2 sections - the painting extraction phase, and the painting matching phase. Each section is broken down further into intermediary steps where combinations of image processing and computer vision techniques are used to achieve the end result of identifying the target painting against a known dataset.

Painting Extraction

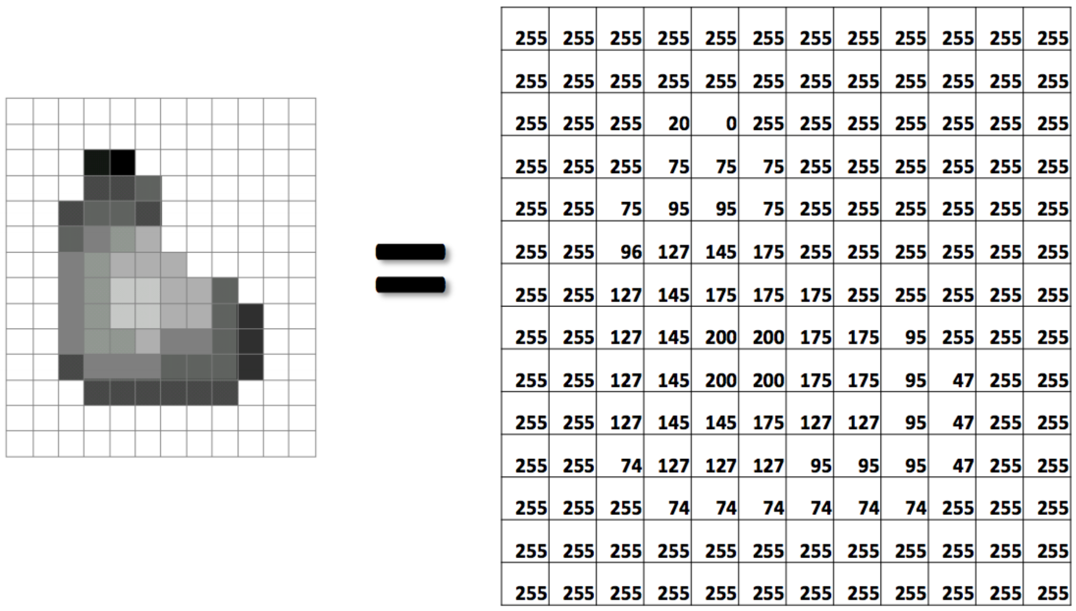
Introduction

Beginning with the painting extraction phase we will look at the 5 step process of extracting the painting from its background. The process involves pre-processing of the image, edge detection, further processing, and finally contour detection and filtering. The final aim is to end up with the extracted painting as our result with the entire background of the image removed in order to increase the chances of the successfully matching the painting to its correct reference image in the matching phase of the application.

The Captured Image

In order to understand the various pre-processing steps required to extract the painting we must first understand what exactly the nature of the data is that we are processing which in this case is an image.

For our purposes an image is simply a grid or matrix of different intensity values. For example a grayscale image may have intensity values represented between 0 and 255 where 0 represents black and 255 represents white. All of the values in between these values make up the different shades of grey.



255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	20	0	255	255	255	255	255	255	255
255	255	255	75	75	75	255	255	255	255	255	255
255	255	75	95	95	75	255	255	255	255	255	255
255	255	96	127	145	175	255	255	255	255	255	255
255	255	127	145	175	175	175	255	255	255	255	255
255	255	127	145	200	200	175	175	95	255	255	255
255	255	127	145	200	200	175	175	95	47	255	255
255	255	127	145	145	175	127	127	95	47	255	255
255	255	74	127	127	127	95	95	95	47	255	255
255	255	255	74	74	74	74	74	74	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255

Figure 9 - A simple grayscale image on the left represented in its matrix or grid form on the right [9]

More complex images such as colour images may be represented in different forms such as RGB or HLS which stand for red green blue and, hue luminance and saturation respectfully. In the case of these 3-channelled images, each channel would have its own grid of numbers so in the case of RGB, each pixel would have a corresponding red value, green value, and blue value.

By manipulating and changing these values across the image it enables us to process them in different ways, some of which will be explained in the following pages in order to extract the painting successfully.

Gaussian Blur

In order to remove any unwanted small edges and image noise produced by the smartphone's camera's sensor due to poor illumination and temperature changes, the captured image is initially pre-processed using a Gaussian blur. The aim of this pre-processing is to allow for better results in the edge detection part of the painting extraction process where we'll be looking to find the edges of the rectangular frame of the painting itself. Image noise negatively affects the results of edge detection techniques as false detection of noise perceived as edges is likely to occur. The goal is to essentially reduce the effects of the noise on the edge detector before running it over the image.

A Gaussian blur (also known as Gaussian smoothing) is achieved by convolving an image with a Gaussian function.

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Figure 10 - Gaussian blur in two dimensions [9]

The above equation expresses a Gaussian blur in two dimensions. It is the product of two different singular Gaussian functions in the x and y directions. In the equation, x is the distance from the origin in the horizontal axis, y is the distance from the origin in the vertical axis, and σ is the standard deviation of the Gaussian distribution.

A Gaussian blur essentially acts as a low-pass filter, passing signals or in the case of images, pixels with intensities lower than a certain cutoff intensity and attenuating pixels with intensities higher than the cutoff intensity.

The result of this formula produces a Gaussian distribution whose values are used to construct a convolution matrix or kernel which is essentially just a grid or matrix of numbers.

Kernel convolution is a process whereby a kernel is passed over an image transforming the image based on what the numbers inside the kernel are. Depending on what those numbers are it can be used for anything from sharpening, blurring, edge detection, dilations, erosions etc.

In the case of the Gaussian blur the convolution kernel weights are themselves Gaussian in order to achieve a smooth blurring effect between the pixels.

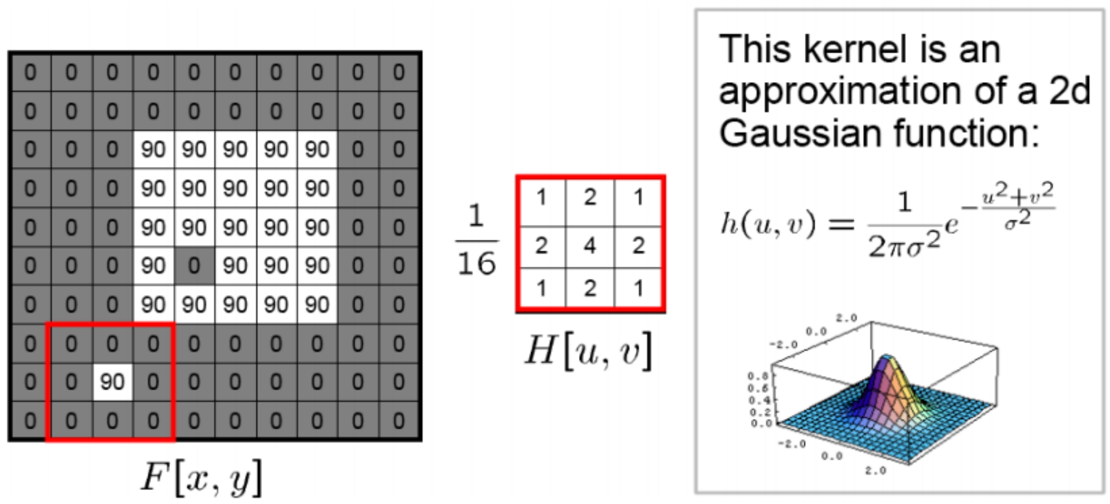

F[x, y]
1/16
H[u, v]
This kernel is an approximation of a 2d Gaussian function:
h(u, v) = 1/(2πσ²) e^(−(u²+v²)/σ²)

Figure 11 - Example of a Kernel whose weighed values are Gaussian being applied to an image [9]

The Gaussian blur results in smooth blurring of the pixels and avoids the blocky nature of a mean blur, the comparison of which can be seen in the below image. The below image also shows the effect the kernel will have on groups of pixels. As we can see in the left hand kernel for the mean blur, the change in pixel intensity is sharp whereas in the case of the Gaussian kernel, the change in pixel intensity is more gradual lending to its smoother appearance.

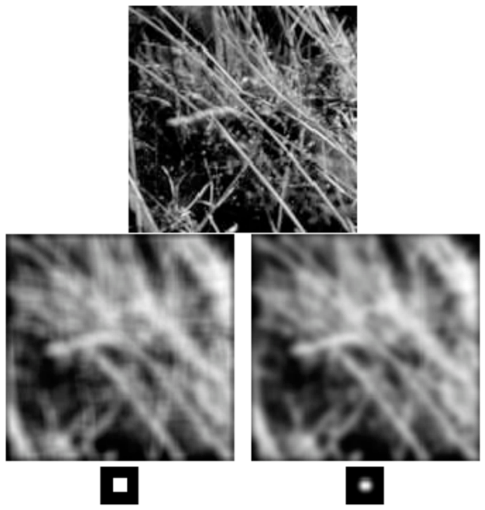


Figure 12 - Mean vs. Gaussian Blur [9]

Canny Edge Detection

In order to detect the outline of the painting's frame that we wish to extract we use an edge detector to threshold the image into black and white pixels where the white pixels represent an edge in the image and black pixels represent where there is no edge in the image.

Thresholding simply involves taking a grayscale image, setting a threshold intensity, and traversing through all of the pixels in the image setting their intensity value to either 0 (black) or 255 (white) depending on whether they are above or below the given threshold. This allows for the creation of binary images which is what we'll end up with after running the Canny edge detector over our input image.

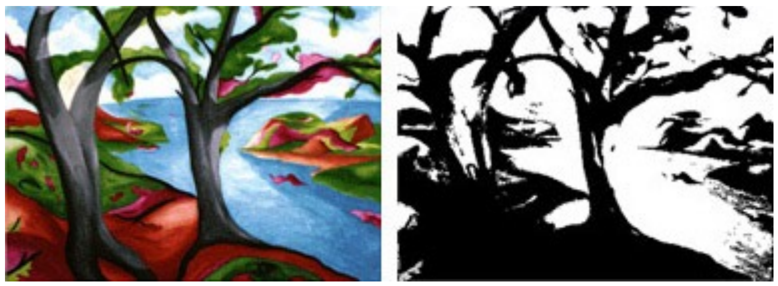


Figure 13 - Thresholded image of a painting

Edges are defined as the points in an image in which pixel intensity changes sharply. The detection of these edges can be achieved through the use of mathematical methods to identify the points in the digital image where they occur. The general aim of edge detection is to extract all of the useful structural information we're interested in from an image and to significantly reduce the amount of data to be processed.

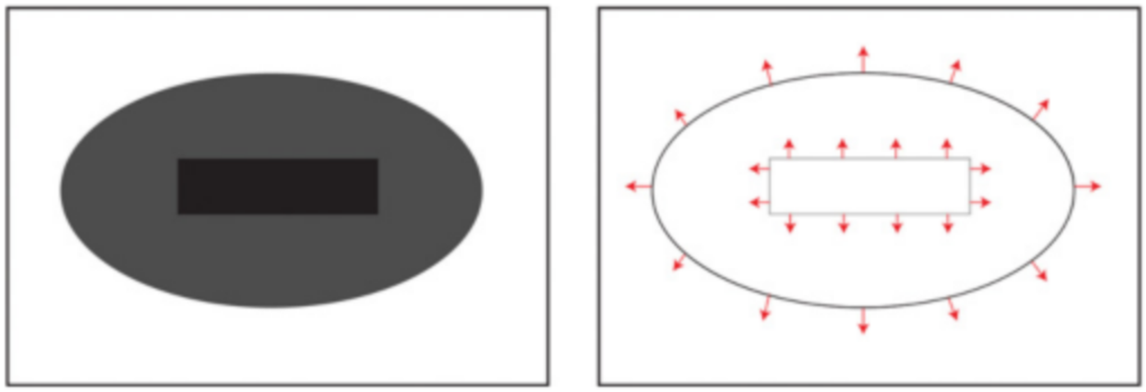


Figure 14 - Basic image on the left, edges and orientations highlighted on the right [13]

Edge detection is usually performed on grayscale images as we only care for the change in pixel intensity in order to detect edges. Looking at the grayscale image below from the OpenCV documentation, we can clearly see with our own eyes were an edge or area of pixel intensity suddenly changes.

Figure 15 - Grayscale image highlighting are of pixel intensity change or edge [12].

In order to express these changes in pixel intensity we use derivatives where a high change in gradient indicates a major change in the image in terms of pixel intensity.

The graph below represents a change in pixel intensity in a 1 dimensional image

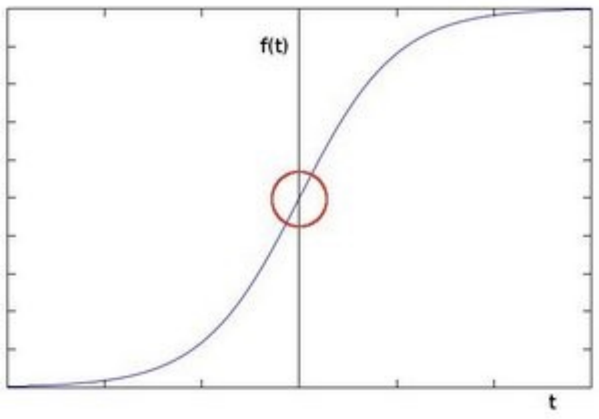


Figure 16 - Change in pixel intensity for a 1 dimensional image[12].

Taking the derivative of this allows us to see this change more clearly. Using this we can detect where edges in an image may exist by finding pixels where the gradient is significantly higher than its surrounding neighbours.

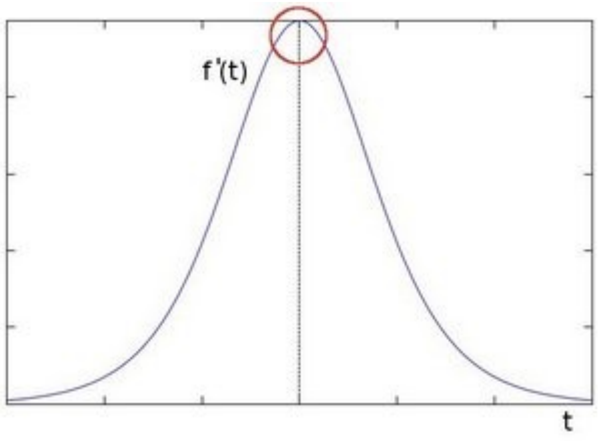


Figure 17 - First derivative of change in pixel intensity for a 1 dimensional image [12].

From calculating the first derivative of the pixel intensity in both the x and y directions we can then find the edge gradient and direction of the edge using the following the equations.

$$\mathbf{G} = \sqrt{{\mathbf{G}_x}^2 + {\mathbf{G}_y}^2}$$

Figure 18 - Calculating the edge gradient

$$\mathbf{\Theta} = \operatorname{atan2}(\mathbf{G}_y, \mathbf{G}_x)$$

Figure 19 - Calculating the direction or orientation of the gradient

In visual form, the edge orientation is represented using a colour to represent what direction the edge is facing.

One of these mathematical methods which utilises the above principles is the Canny edge detector which was published in 1986 by John F. Canny.[18] It aims to have a low error rate in detecting edges meaning we only want to detect edges that actually exist, good localisation meaning the distance between edge pixels detected and real edge pixels have to be minimised, and to have a minimal response meaning we only want to detect each edge once.[11] We use the Canny edge detector here to try and detect the structural element of the painting's frame.

The process of the Canny edge detector can be broken down into 4 different general steps:

1. Apply a Gaussian filter to reduce noise and smooth the image (Gaussian blur explained in previous section)
2. Detect the gradient intensities in the digital image.
3. Apply non-maximum suppression to remove pixels that are not considered to be a part of an edge.

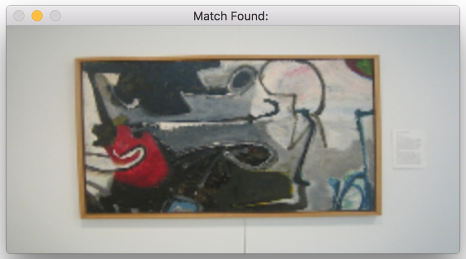
Non-maximum suppression is a technique to thin the edges we have found in the image. It will essentially preserve the part of the edge where only the sharpest change in intensity value occurs in order to remove blurred edges in our edge detected image. It does the via the following algorithm:

- The edge strength in both the positive and negative directions are compared

- The value of the pixel's intensity will be preserved if its edge strength is larger than the other pixels in the mask with the same orientation, otherwise it will be suppressed.

4. Apply hysteresis to remove any undesirable edges caused by noise and colour variation that do not actually represent a real edge in the image.
    a. If a pixel gradient is higher than the upper threshold, accept the pixel as an edge
    b. Reject pixels whose gradient value is below the lower threshold
    c. Pixels whose gradient is between that of the lower and higher threshold are only accepted if they are directly connected to a pixel whose gradient value is above the upper threshold. This basically allows for strong continuous edges that may taper off at some point to be preserved. Taking the example of the below painting we may see that one part of the frame produces stronger edges than the other side because of illumination changes. This step in hysteresis will preserve the weaker edge of the frame because it is connected to the stronger one by basically saving the edges that are connected to the edges above the top threshold.

The process of hysteresis will essentially remove all weak edges that are not connected to strong edges.
The resulting image should be a thresholded image showing the edge pixels in white and every other pixel in black as in the example below.



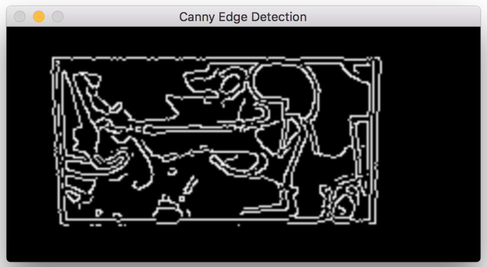


Figure 20 - Canny edge detection performed on a painting

Dilation

Having detected the edges in the image of the painting hanging on wall, you may notice that while we can see the outline of the frame there doesn’t exist a consistent rectangular edge or contour around the whole frame. This is due to the fact that not all of the edges will be detected correctly due to factors such as the Gaussian smoothing blurring out the corners and junctions in the edges making them harder to detect by the Canny edge detector and other issues such as poor illumination conditions etc.
In order to rectify this we use what is known as a dilation to fill in these gaps in the edges and connect up all of the broken junctions to allow for the frame to be detected in its entirety.

A dilation is a morphological operation. Morphological operations are a set of operations that process images based on shapes called ‘structuring elements’ in order to generate an output image from a thresholded image. The structuring element is essentially a kernel or grid of numbers with an anchor point which is normally the centre of the structuring element.

Dilation is a technique for expanding the number of object pixels[13].
In the case of dilation the maximum value of the pixels contained within the structuring element or kernel are found and the anchor point’s pixel value is replaced with this value causing the bright pixels of the edges to dilate and become more defined, helping to join broken edges, and fill in gaps between larger edges in the image.

We use dilation here following the Canny edge detector to fill in and connect any broken edges that may have not been correctly detected. This increases the chances that the edge of the frame of the painting will be detected allowing it to be extracted later on.

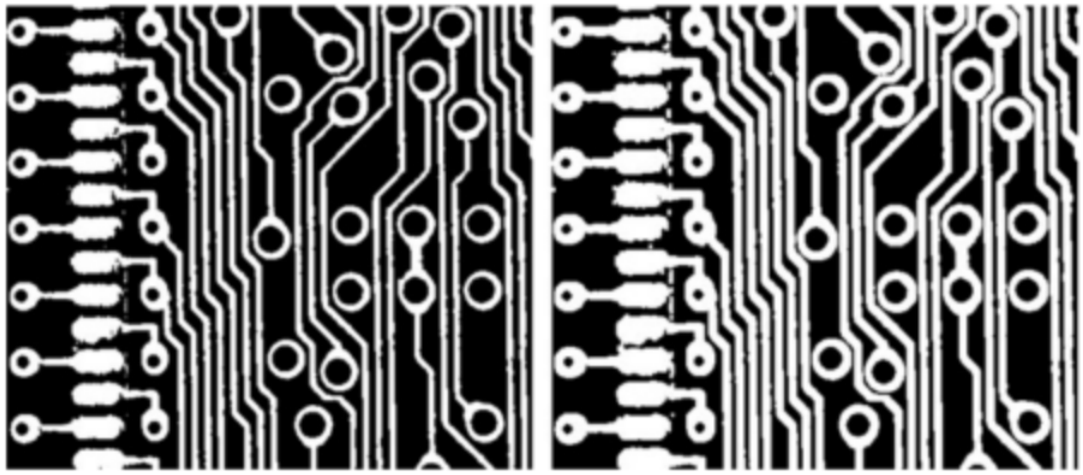


Figure 21 - Binary image of a circuit board following a dilation operation [13]

Finding and Filtering contours

A contour is a line or curve connecting continuous points along a boundary that either have the same colour or intensity. Contours are a useful for shape analysis and object detection and recognition. We use contours here to describe the area in which we believe a painting may exist in the given input image after the previous processing has complete. For the best accuracy contour detection should be used on binary images. Since we've used Canny, our image is already thresholded. The edges found within the image will also be connected up from the previous dilation step. Contours will be found along the white lines in the image that are continuous. In order to save some memory we use the contour approximation method. This method avoids storing the (x,y) coordinates of the boundary of a shape by just storing the start and end point of each straight line. A coloured line and then be drawn between these points to represent the contours.

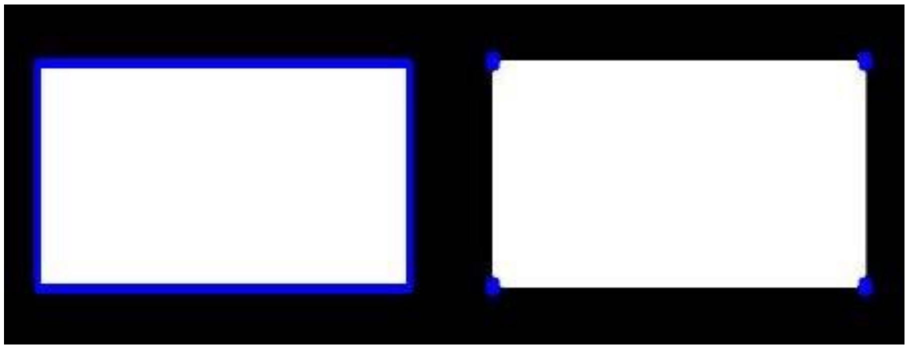


Figure 22 - Image on the right depicts using no contour approximation, image on the left using contour approximation method [16]

In order to find the contour representing the frame we only wish to detect the external contours. To achieve this we must look at the hierarchy of the contours.

Finding contours is normally used with the goal in mind of detecting some kind of an object in a given scene. Objects will be in different locations in the scene, but in our case, shapes will be found contained in other shapes - the painting and its elements in the painting frame.

Considering the image below we see some arbitrary shapes labeled 0-5 where 2 and 2a represent the external and internal contours of the outermost box.
The contours we're interested in in terms of finding our painting's frame are the most external ones - so in this example contours 0, 1, and 2. The hierarchy level they are described as being a part of is called hierarchy-0.

Using the 'RETR_EXTERNAL' flag in OpenCV when detecting contours allows us to filter out all contours that are not extreme outer flags i.e the child contours. Looking at the below example we only consider the contours at the hierarchy-0 level, contours 0, 1, and 2.

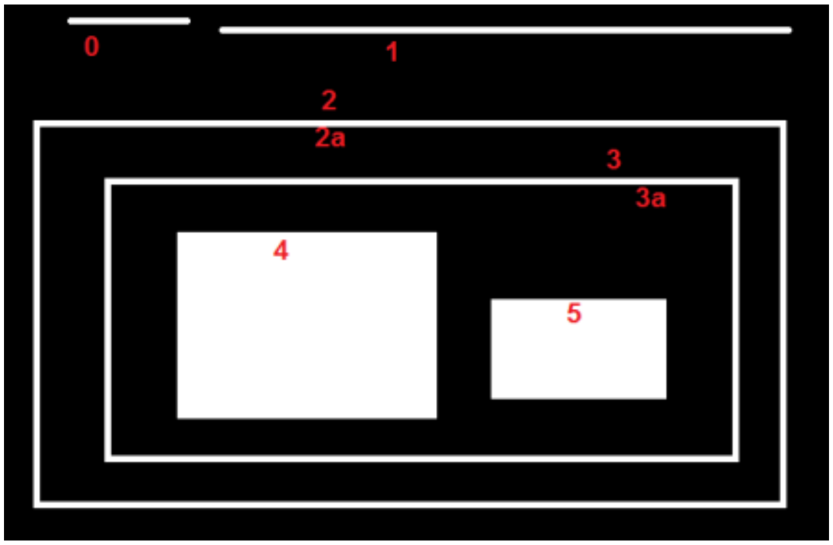


Figure 23 - External contour hierarchy representation. [15]

Using the above mentioned flag we return only contours 0, 1, and 2 and filter out the rest of the child contours we're not interested in as they are likely not to be the frame of our painting.

In order to find the contour representing the painting from these remaining contours, we filter the contours by area, selecting the contour containing the largest area which is most likely to be our painting's frame. The coordinates of this contour are then used to extract the painting from its background using OpenCV's crop function.

Painting Matching

SIFT (Scale invariant feature transformation)

SIFT was developed in 2004 by David Lowe[19] with the aim to compute robust features in digital images for tracking and recognition.
In order to match the extracted painting to its correct reference we use SIFT to detect key-points and create SIFT features that we can then use in a later stage to match them correctly to their reference painting. SIFT is useful in this case as it is invariant to images of different scales and rotations as well as partially invariant to viewpoint and illumination changes which will be the case here as users will be photographing the target painting from different angles and distances than that of the reference painting we wish to match it to.

SIFT is quite a complex algorithm involving many parts and concepts. It can be generally broken down into the 5 following steps:

1. Constructing a scale space - Internal representations of the original image are created to ensure scale invariance.

2. Laplacian of Gaussian approximation

3.
    a. Key-points detection - Locate the local maxima and minima in the difference of Gaussian to locate the key-points
    b. Removal of poorly detected key-points - Remove key-points detected at edges and low contrast regions.

4. Assignment and orientation of the detected key-points - Calculate each key-points's orientation.

5. Generation of SIFT features

Constructing a scale space and Laplacian of Gaussian approximations

Objects we view are only meaningful at certain distances. If you view an apple at 1m distance from yourself it is still perceived as an apple to you. However if you view the same apple 10 kilometres away it won't exist as far as you can tell. Scale spaces attempt to replicate this concept in digital images.

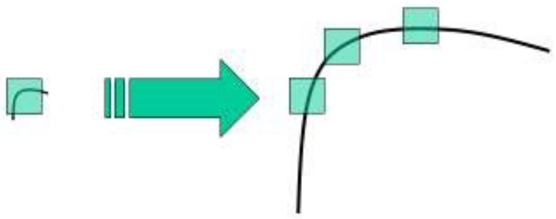


Figure 24 - Another example of scale variance, the corner will not be perceived as a corner when zoomed in.[14]

In order to create a scale space for the extracted painting, the image is taken and a series of progressively more blurred out images are created using Gaussian blur, the original image size is halved and the blurring process is conducted again. The number of times this happens is determined by the number of octaves specified. Lowe suggests that 4 octaves and 5 blur levels are generally ideal for use of the algorithm.

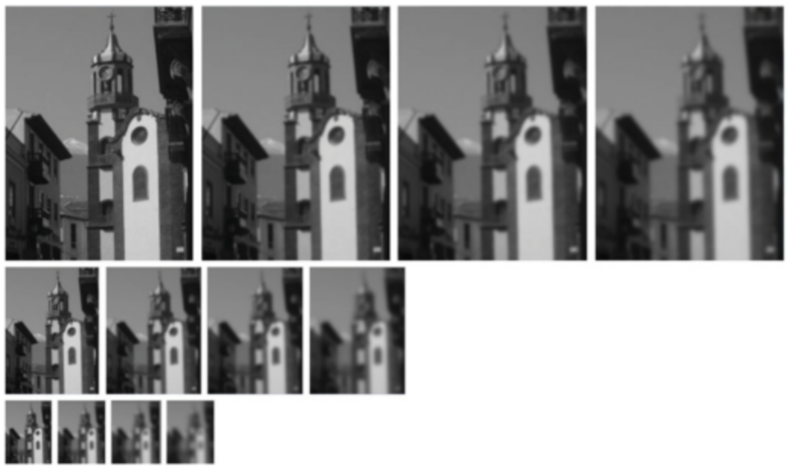


Figure 25 - Three octaves of scale space for the image of a clock tower[13]

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

- L is a blurred image
- G is the Gaussian Blur operator
- I is the image
- x, y are the location coordinates
- σ is the "scale" parameter. Think of it as the amount of blur. Greater the value, greater the blur.
- The * is the convolution operation in x and y. It "applies" gaussian blur G onto the image ‘I’.

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

The actual Gaussian blur operator used in SIFT.

Following on from where we created the scale space of the image, we now use those blurred images to generate the DoG images which are useful for locating key-points.

In this step we take those images, blur them again and calculate the laplacian to locate edges and corners which can be used for finding key-points. The blur is used to smooth out noise generated from calculating the laplacian. While this works, calculating the laplacian is computationally expensive so an approximation is taken instead by finding the difference between two different scales instead to find potential stable key-points locations. See below figure for an example.

(DoG)

$$D_n(i, j, k, \sigma) = L_{n+1}(i, j, k, \sigma) - L_n(i, j, k, \sigma)$$

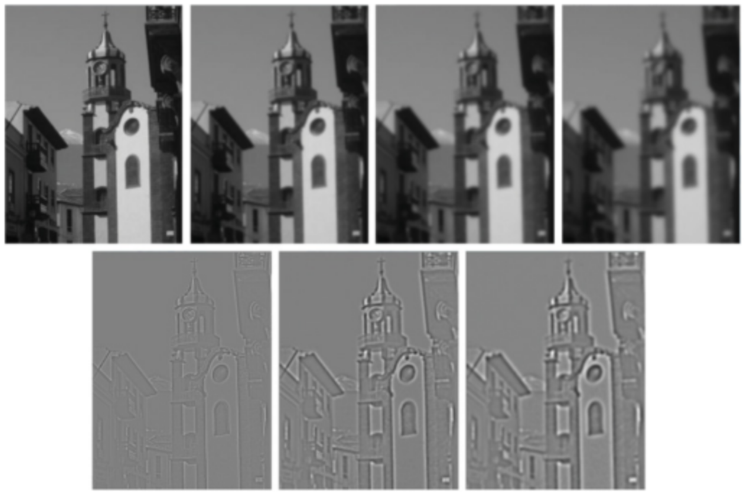


Figure 26 - Above: Gaussian smoothed images for a certain scale, Below: Difference of Gaussian images [13]

Key-points detection and filtration

In order to find key-points we need to locate the maxima/minima in the difference of Gaussian images and find the sub-pixel maxima/minima.

In order to locate the maxima/minima in the difference of Gaussian images, we iterate through each pixel and check its neighbours in both the current image and the image above and below it at the different scales.

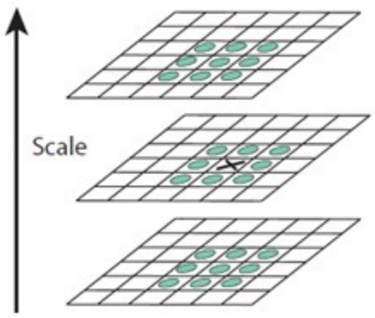


Figure 27 - Considering pixel X and its neighbours (green circles) at different scales for an image.[14]

It’s worth noting that the we do not locate key-points in the uppermost and lowermost scales as there are not enough neighbours to compute the comparison.

Some of the key-points detected this way will be found on edges in the image and some will be of low contrast. These key-points are not suitable for matching so we have to filter them out. To remove key-points of low contrast the magnitude of the intensity for the pixel is checked and if it is below a certain threshold then the key-points is removed.

In order to remove edges we calculate the gradient in the x and y direction at the key-points. In the case of finding an edge, one of the two gradients will be much larger than the other. The other possible outcomes is that we’ll come across a non-edge region where the gradients will be smaller or a corner where both gradients will be large. This is calculated mathematically using a Hessian matrix at the location and scale of the key-points similar to that of Harris corner detection.

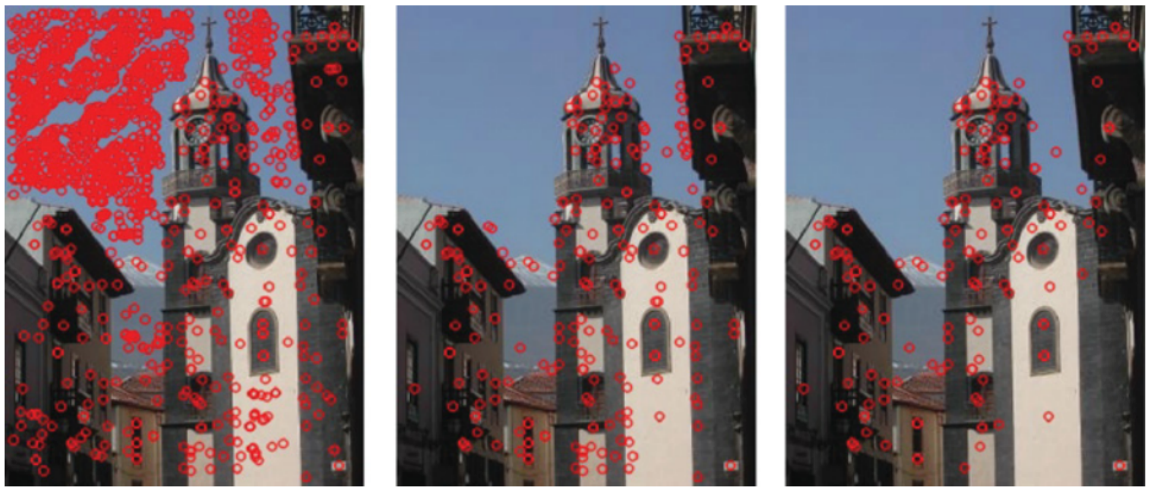


Figure 28 - Example of key-points filtering from contrast and edge tests [13]

Following these tests for contrast and edges we now have a set of actual key-points that we know are stable and scale invariant as it will be the same as the scale in the blurred images.

Assignment of orientation

The next step in the process is to assign orientations to each of the stable key-points we now have in order to achieve rotation invariance. This is achieved by finding the direction and magnitudes of the gradients around each of the key-points and assigning the most prominent orientation to each key-point. All subsequent calculations are then done relative to this calculation ensuring that the rotation invariance is achieved.

The following two formulae are used to calculate the respective magnitude and orientations of each pixel in each key-points in the image:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \quad \textbf{(magnitude)}$$

$$\Theta(x,y) = tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y))) \quad \textbf{(orientation)}$$

(L(x,y) represents an image sample)

These values are then mapped to a histogram. The histogram is broken up into 36 different bins where each bin represents 10 degrees. Each key-points’s proportional magnitude is added to the orientation bin it falls into. This is done for all pixels around a key-points resulting in a peak in the histogram representing an orientation that the key-points are then assigned. Any additional peaks above 80% of the highest found peak in the histogram will also be used a new key-points. In this case a new key-points made with the same scale and location as the original but with this lesser peak as its orientation. About 15% of the original key-points will result in multiple orientations.

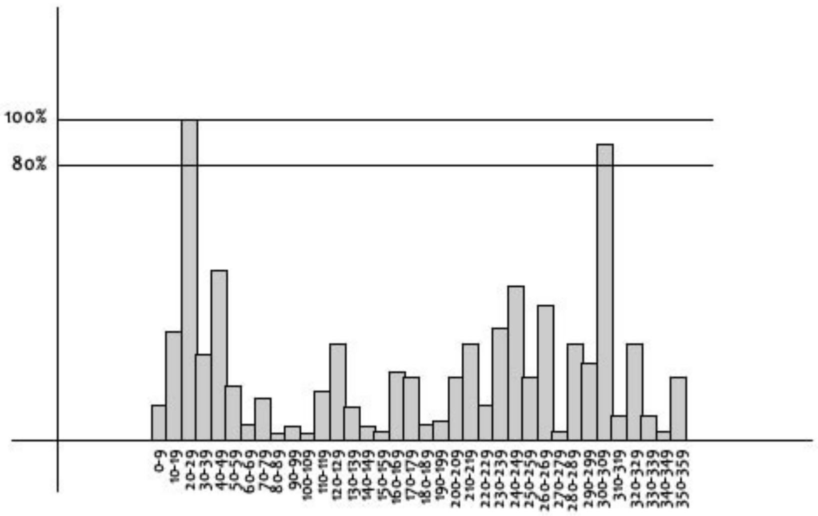


Figure 29 - Example histogram for representing the proportional magnitude of each orientation across pixels surrounding a particular key-points

To summarise, a histogram of the most prominent pixel gradient orientations is used to determine the orientation of an individual key-points in order to achieve rotation invariance.

Generate SIFT features

Now that we have both our scale and rotation invariant stable key-points we now want to make descriptors for these key-points in order for to be able to identify and compare them with the other key-points. To do this a blurred image at the closest scale is selected for a particular key-points. We then select sampling points around the key-points and compute their individual gradients and orientations. The image is then rotated by the key-points orientation so that all orientations are normalised with respect to the key-point's orientation. We then divide up the surrounding region of the key-points into four different subregions and compute a weighted histogram of the orientations similar to that of the previous step by gradient and location for each of the subregions.[13]

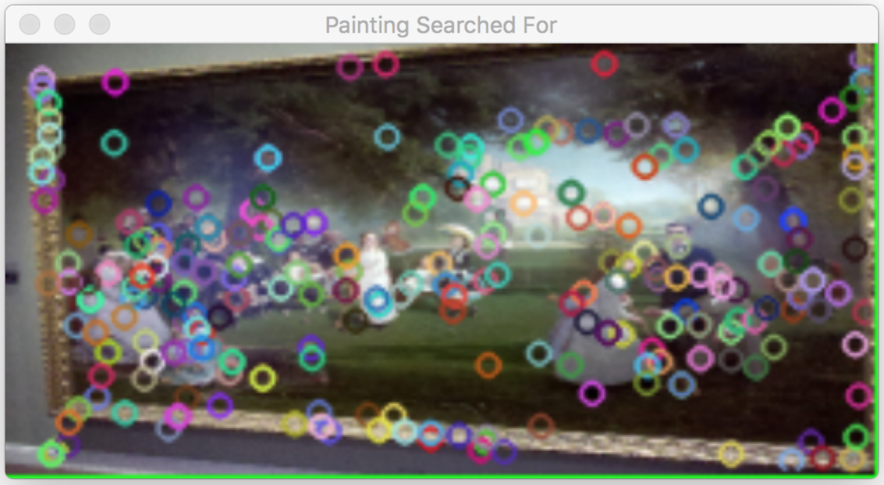


Figure 30 - SIFT features of a painting

K Nearest neighbour feature matching

Now that we have our key-points and descriptors for the key-points we can start matching them against those of the known set of reference paintings that are stored on the onboard database. In order to find matches fast and efficiently we use the FLANN based kNN matching algorithm. FLANN stands for Fast Library for Approximate Nearest Neighbours and it contains a collection of algorithms optimised for fast nearest neighbour search in large datasets and for high dimensional features. One of which is kNN which stands for k Nearest Neighbour (*k* being a number).

The main idea behind kNN is to search for the closest match of the test data in feature space.
So in our case for each SIFT feature we look for its closest match in the feature space of the reference painting's detected SIFT features.

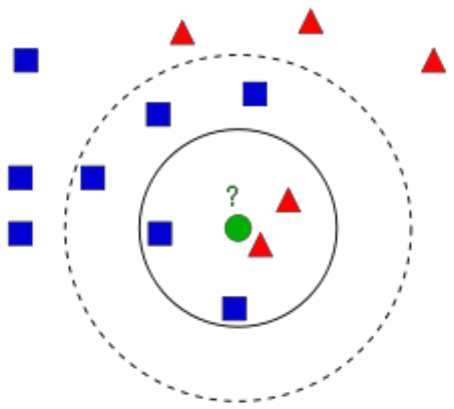


Figure 31 - kNN diagram [17]

Taking the above diagram as an example we see two classes of shapes in a 2D feature space, the blue square and the red triangle. When a new member is added to the feature space we need to decide whether it will become a blue square or a red triangle (classification). In order to decide we have a few options. One option would be to check the green circle's nearest neighbour. From the feature space we can clearly see that in this case that would mean the green circle would become and red triangle - this is nearest neighbour. However the issue with this method is that it doesn't take into account the fact that while red triangle may be the nearest class, the blue squares have more strength in locality in comparison. With kNN we check for the *k* nearest classes instead. Taking an odd number for *k*, say three, we look at the 3 closest classes to the green circle. We see that the three closest classes are 2 red triangles and 1 blue square so it becomes a red triangle. Taking *k* to be five, we would have 2 red triangles and 3 blue squares, making the green circle become a blue square.

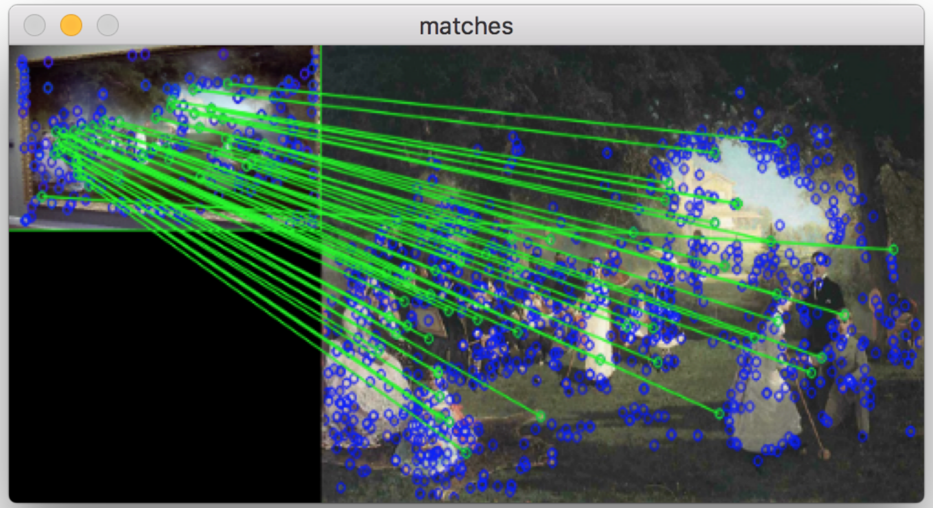


Figure 32 - Matching of SIFT features between and extracted painting and its reference

Summary

This section describes the theory behind the individual components of the application and how they fit together. It examines the 5 stage painting extraction technique that was used to remove the background from the image and isolate the painting, and the matching algorithm comprised of SIFT and kNN matching to match the painting to its correct reference in the on board database. In the next section of this report we will look at how the application performed from both a perspective of successfully matching paintings to their correct reference and in terms of the time it took to find a match.

Testing and Results

In this section of the report we will look at the results produced by the final implementation of the application. This will be broken down into multiple parts discussing the success rate at which paintings were extracted, individual test cases, successful matching rates, and the speed in which it took to typically match a painting to a reference painting. By analysing the results and how each part of the application performed individually we can determine where the strengths and weaknesses lie in the application's implementation, where improvements can be made, and what worked well in achieving the end goals of this project.

In order to test the different components of the application the Stanford Mobile Visual Search Data Set of museum paintings was used. This consists of over 92 different paintings photographed using the Droid smartphone device's camera. Each reference also contains a reference image of a high quality scan or photograph of the painting itself to be matched against.

Painting Extraction

For this part of the results we will first look at two test cases for the painting extraction. One where the painting was successfully detected and extracted, and one where the painting was not correctly detected and extracted and how that affected the result produced in the matching stage of the application's pipeline. For the purposes of this project I am defining a successful extraction of a painting to be when the painting in its entirety is separated from its the majority of its background. If a painting is only partially extracted, missing a sizeable chunk of what makes up the painting then it is considered a non-successful extraction. It is still considered a successful match if small parts of the frame and/or background are still present in the extraction.

Test case 1 - Successful Extraction

The following images illustrate the test case for a successful painting extraction for one particular painting. As we can see from the second image the largest contour detected from the process of Gaussian blurring, Canny, and dilation has managed to capture the outline of the paintings frame. While not perfect in this case due to the detection of the small shadow cast by the frame at the bottom of the painting, it is still good enough for the painting to be successfully extracted from the background of the image when the coordinates, height, and width of the largest contour are used to form a bounding box for extraction. Following extraction this painting was then correctly matched to its corresponding reference painting.

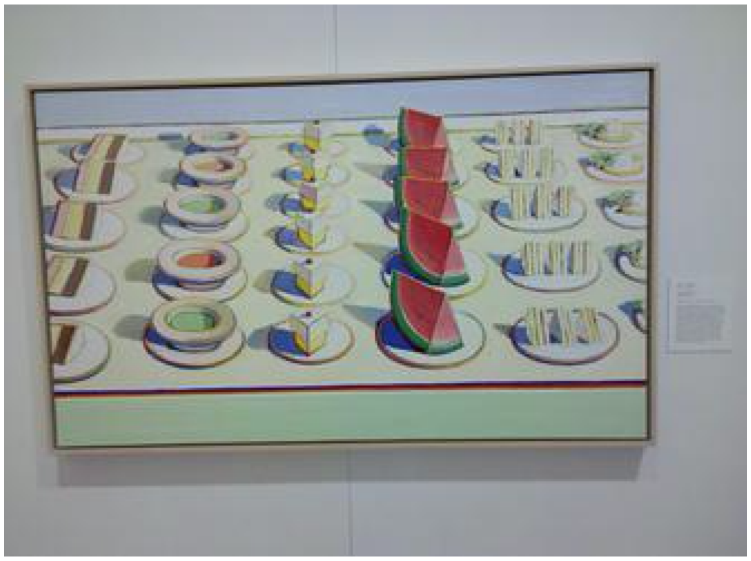


Figure 33 - Original input image of painting captured with user's smartphone

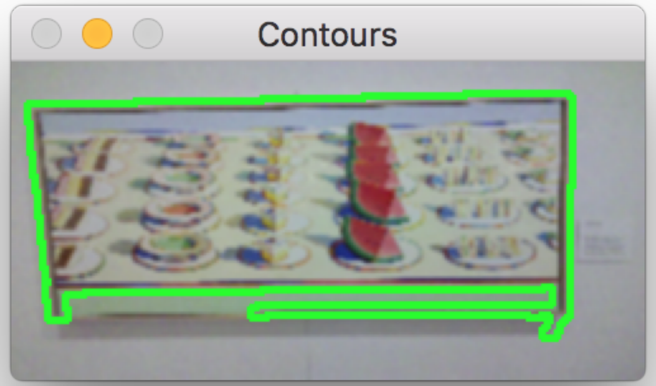


Figure 34 - Largest external contour filtered out from other contours

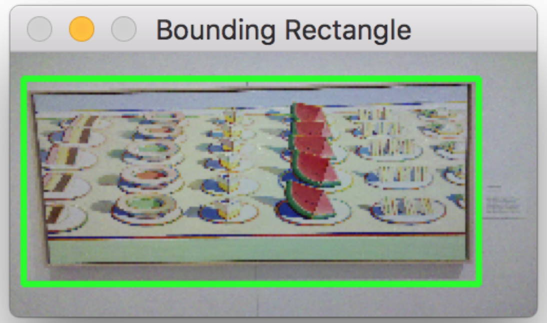


Figure 35 - Bounding box drawn using coordinates of contour with largest area

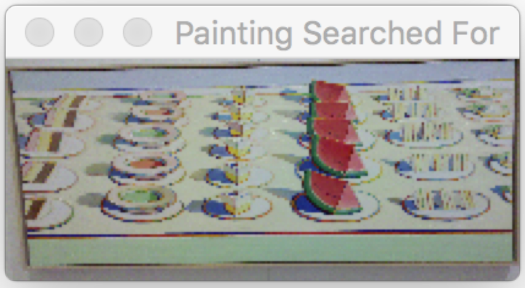


Figure 36 - Extracted painting from image

Test case 2 - Unsuccessful Extraction

In this case we see an example of a painting that has not been extracted successfully as the top half of the painting has not been included in the detection of its border. This is likely due to a few factors. First of which is likely due to the bright glare on the surface of the painting's glass, which is leading to the detection of other edges we don't care about (i.e the edge between where the bright light from the glare and the actual painting itself). The second is that the weaker edges on the left and right hand side of the painting are not being detected by the by the edge detector. As we can see for the bottom two figures, the edges of the frame have not joined up and instead the largest connected contour ends up being the the contour of the bottom part of the frame connected to elements in the painting and the large light of the glare. This results in a bounding box that does not encompass the entire painting and a failure to extract the whole painting from its background.

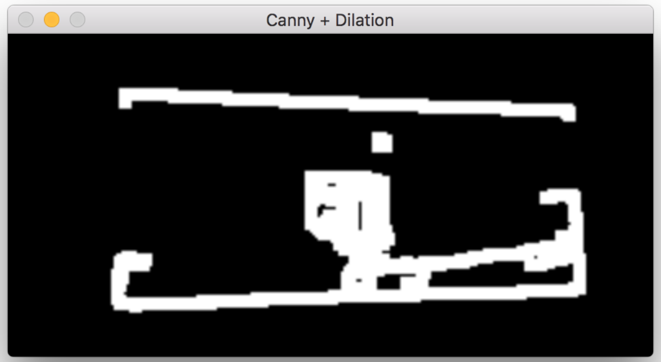


Figure 37 - Poorly detected edges following dilation and Canny

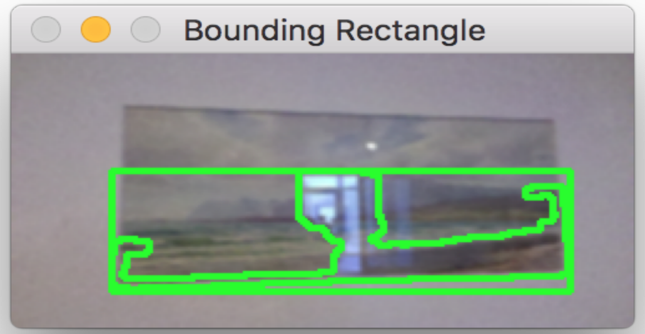


Figure 38 - Poorly detected frame of painting

A way to rectify this and to successfully identify the painting would be to lower the lower threshold of the hysteresis procedure in the Canny edge detector. Lowering it from 100 to 10 results in the detection of some of the more weaker edges that we may wish to detect. Lowering the threshold allows for more edges to fall between the upper and lower thresholds of the hysteresis procedure. These edges will be detected as they are directly connected to a pixel whose gradient value is above the upper threshold.

As in the case in this painting the frame is very subtle making it hard to detect. The images below detail their detection when this lower threshold has been lowered.

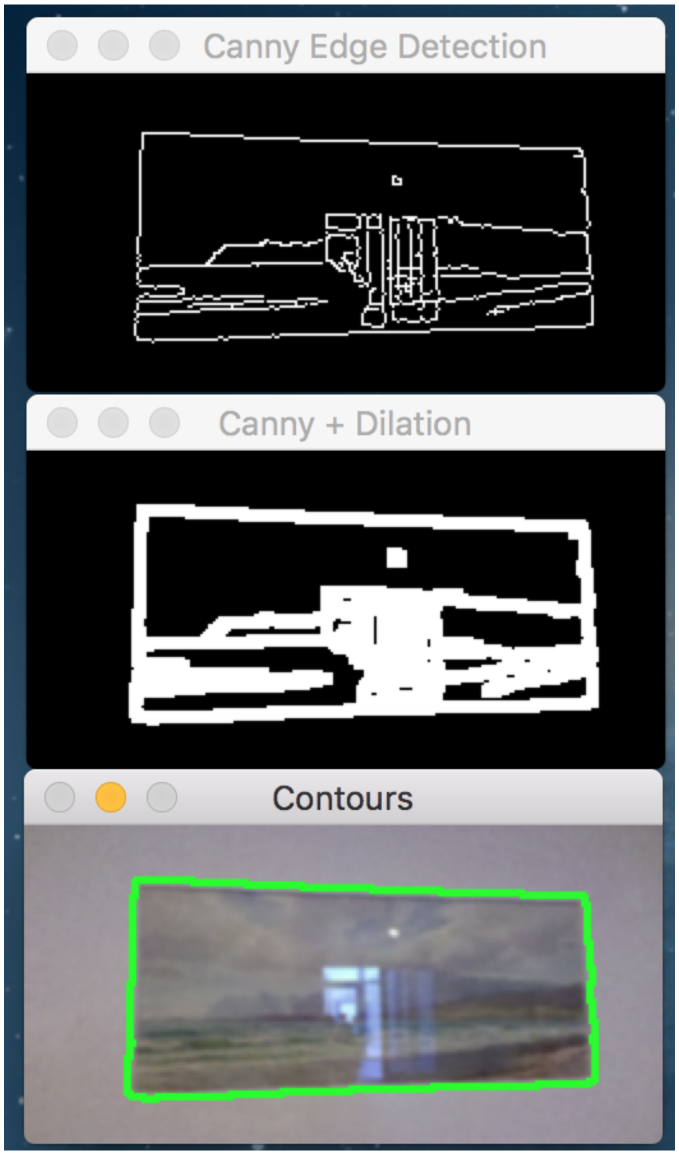


Figure 39 - Improved edge detection of weaker edges by lowering lower threshold of the hysteresis procedure

In the case of this painting, using the original parameters for Canny it was not correctly matched to its correct reference but upon increasing the edge sensitivity and getting a successful extraction it was matched successfully to its correct reference painting.

Another typical downside of the painting extraction technique is the inclusion of some external elements to the painting that we do not care about. While this is negligible in most cases, some of the paintings with the larger more elaborate frames resulted in the extraction of large portions of the scene we don't care about which in turn negatively affects the accuracy of the matching for paintings with those particular frames.

Take the example below of the following painting with the large golden frame.

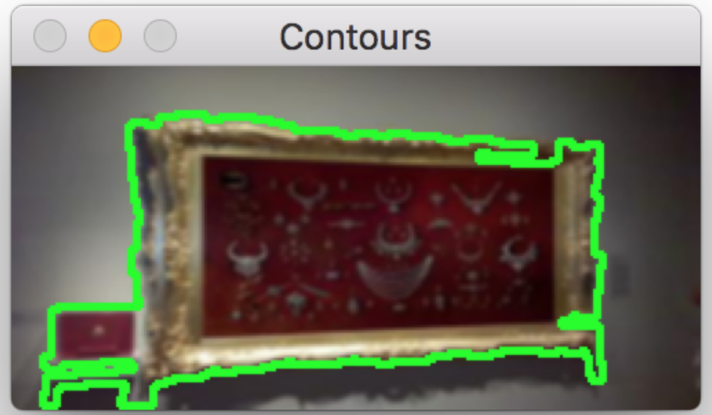


Figure 40 - Poorly detected painting

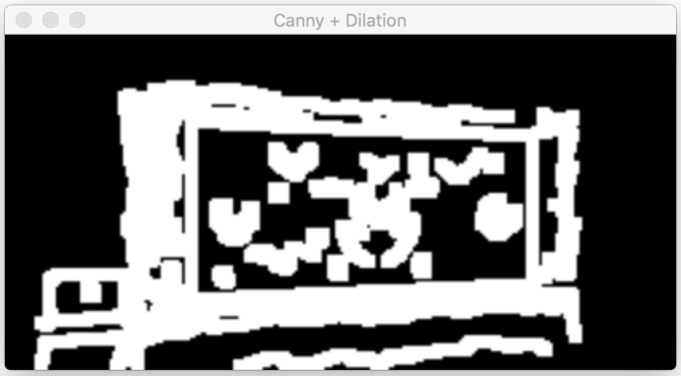


Figure 41 - Canny edge detection and dilation

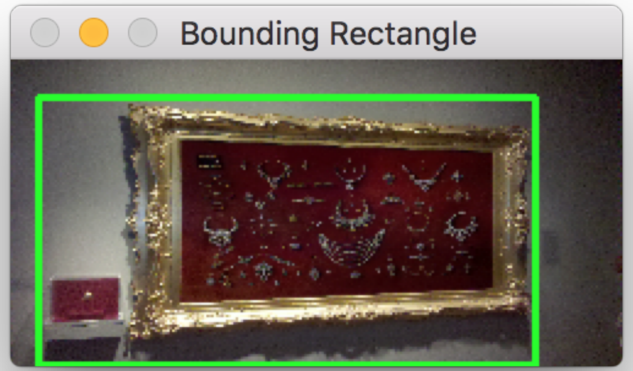


Figure 42 - Poorly detected painting in extraction phase containing too many external elements

In this case the combination of the large elaborate frame and dark shadows caused by poor indoor illumination has led to the detection of edges and dilation of those edges to connect together elements in the scene that are not actually connected which has led to the incorrect extraction of the painting. In this case features that will not aid the matching phase will be detected and interfere negatively with the matching algorithm. In this particular case the painting was not matched correctly and no strong match was found. In order to rectify this one could increase the threshold required for edges to be detected in order to not detect these external non-existent edges.

From looking at these two failure cases we can see how tweaking the various parameters of the various components of the application can result in more favourable results for particular test cases. However, looking at the big picture these changes do not always bode well for the overall results of the application, often lowering them to a degree (for instance, increasing the edge sensitivity of Canny resulted in a drop from 58% success rate to 49%). One of the downfalls of an application such as this is the fact that you can spend an endless amount of time tweaking different parameters of different parts of the application and testing to see what the optimum combinations are in terms of both efficiency and performance.

Overall results for painting extraction

Overall the results the painting extraction phase produced were in most cases quite usable for the matching phase. While many paintings did contain some external elements, it was found that 90% were extracted successfully, with only 10% of extractions missing significant parts of the painting. However of those 90% were the painting was extracted in its entirety there was usually some external elements included as well. Some of which were negligible but some extractions contained external placards, shadows cast by other paintings, and other smaller objects in the scene that we would not wish to detect.

Conclusion

In this section of the report we looked at two typical test cases for the painting extraction technique developed for this application and how it performed in a typical successful extraction case and in two cases where it failed and why. From the failure test cases we can see where the technique falls short and some potential areas for improvement. Finally looking at the overall results we can see that the extraction technique worked quite well for extracting the painting from its background but often included some external unwanted features in this extraction.

Painting Matching

In this section of the results we will look at the success rate of the painting matching technique using SIFT and kNN matching to match the SIFT features between the extracted painting and its reference. It was found that a painting was more likely to matched correctly following a successful extraction from the background of the image. This was due to the fact that SIFT features of external elements were not detected and either not enough matches were found or some of the SIFT features were incorrectly matched to SIFT features calculated in the reference paintings.

It was also found that the overall success rate for matching paintings to their reference following the extraction and matching phase (i.e the entire pipeline of the application) was 58%. In total, 53 of the 92 paintings were correctly matched to their reference. In order to evaluate the matching technique used (SIFT + kNN) we will again look at some typical successful match cases and some unsuccessful match cases. By doing this we will hopefully be able to identify the strengths and weaknesses of the technique and where potential improvements could be made in order to improve the procedure and increase the match rate.

Test case 1 - correctly matched extracted painting to reference

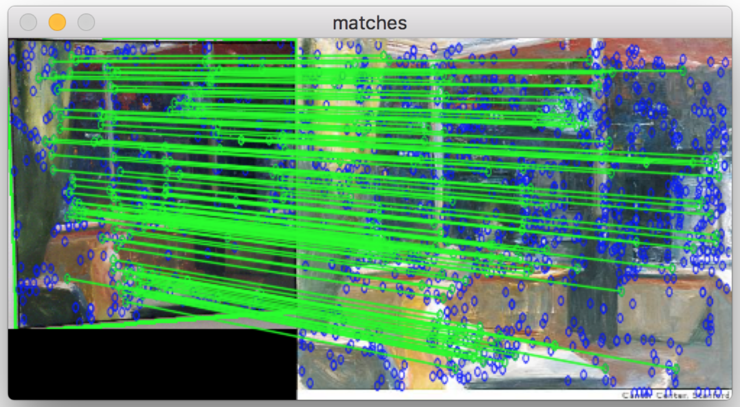


Figure 43 - Example of a successful match

In the above example we see a typical case where an extracted painting has been correctly matched to its reference. Numerous SIFT features have been found and matched between the two paintings far surpassing the set threshold of 10 matched features in order to trigger a strong match in the application. This is likely due to good extraction of the target painting whose viewpoint and perspective is quite similar to that of the reference image. The scene is also quite well illuminated, with no artefacts such as shadows or glares to interfere with the matching technique.

Test case 2 - incorrectly matched extracted painting to reference

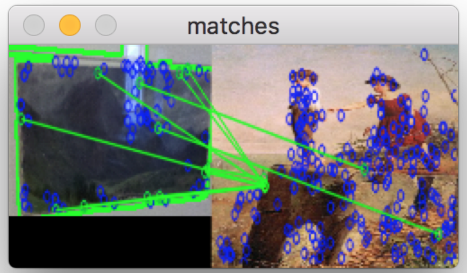


Figure 44 - Example of an unsuccessful match

In the above example of the unsuccessful match we can see that a number of SIFT features have been found in the extracted painting. In this case a strong match surpassing the threshold has not been found so the best possible match has been returned by the application. Many of the key-points found here have been incorrectly matched to one particular key-point in the reference image. This is largely due to the poor illumination of the painting itself in the photograph, the large glare on the paintings frame, and poor detection of good SIFT features for matching, all of which have been incorrectly matched.

Average time taken to find a match

It was found that the average time taken for a painting to be matched to a reference painting was 3.34 seconds. This slower than expected matching time is due to two main bottlenecks identifiable in the application. The first being that the SIFT features, key-points, and descriptors of each reference painting should be serialised in order to avoid having to calculate them each time for each reference every time we wish to identify a painting. This is explored in more detail in the 'Future work' section of this document. Essentially when an image is captured and the painting extracted, in order to match it against a reference painting, SIFT features must be calculated for every reference painting it is compared to which takes significantly more processing time than it would take if this data was serialised beforehand.

The second bottleneck is to do with paintings who fail to find what is known as a ‘strong match’. In this application any extracted painting and reference that when compared using kNN matching to have more matching SIFT features than the set threshold of 10 is considered a good match and returned instantly. Failing to find a strong match, the application will loop through all of the reference paintings and keep track of the painting with the most matching key-points. While sometimes successful, this is usually the case when a painting will not be successfully matched to its reference for one reason or another and is another big part of the reason why the average painting match time was higher than expected. In some cases it can be found that to match a painting for which a strong match is not found may take up to ~4.5 seconds.

The graph below illustrates how long it took in seconds for each painting in the dataset to be matched to a reference painting. Note that not all of these matches are successful matches i.e that the extracted painting was matched to its correct reference.

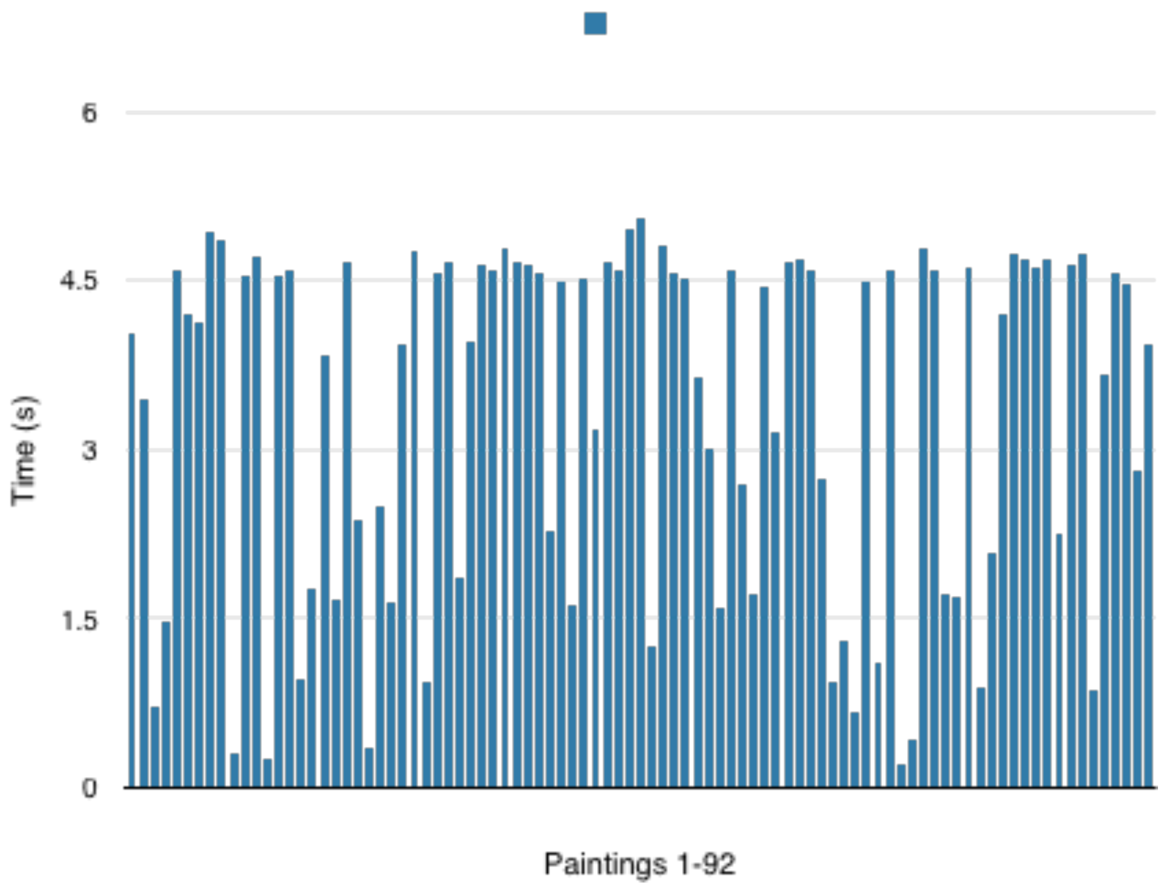


Figure 45 - Chart of time taken to match each painting in the dataset

Future work

The application itself is not fully finished and there are a number of improvements and additions that were to made in order to improve the accuracy, efficiency, and presentation of the final program.

Serialisation of reference painting SIFT feature data

In order to increase the overall performance of the application the descriptors and key-points for the reference paintings could be pre-computed and stored in the onboard database. This would improve the speed at which the reference painting is compared to the references in the matching stage of the application as it would no longer have to compute SIFT features for each reference painting each time a request to match a painting is made. It would also improve the scalability of the application as it would no longer increase in time complexity as the dataset of reference paintings gets larger and larger resulting in more computations in finding SIFT features. This was attempted during the duration of the project but was unfortunately unsuccessful due to storing issues using SQLite.

Categorisation of Paintings

Another way to increase the efficiency and speed of the application would be to implement another step to the matching phase whereby the reference paintings would be categorised by some attribute prior to any matching. For example, the paintings could all be divided up into categories based on how much of the colour red was in each painting - ‘low’, ‘medium’, and ‘high’ amounts of red. Then when a painting is extracted the application could check the level of red in the painting and only match against paintings in the corresponding category, ruling out the other two.

Other Improvements

Improvements and additions can be added to the application in terms of the app's UI and design. The implemented UI was quite a basic layout of buttons and image views to be filled in with the relevant extracted and matched paintings. A more appealing and responsive material design could have been implemented in order to improve the application in this respect. Actual information of the paintings contained in the app's onboard database must also be added for it to be useful in gaining more information about them such as the painting's title, artist, year etc.

Another feature that was originally planned for the application was that of GPS tracking to allow for the identification of non-painting related artworks such as statues that would be located in a museum or art gallery space. In order to identify them the application would grab the device's current GPS coordinates in order to determine which piece of artwork the user was looking at.

Finally in order for the application to be usable in different art galleries and museums there would be options to download the reference paintings for a particular gallery built into the app. Similar to an in app purchase, the galleries themselves would be able to provide the reference paintings and their respective data available to download via the app where users could pull it down and then use it as an electronic guide for their particular establishment.

In summary, much of these additions to the application were either attempted at some point or were not feasible given the time frame of the project itself but would have been implemented given a larger time frame. Without them the application is still quite functional however they would have improved it on both a performance and appearance level if implemented.

Conclusion

The main goals of this project were met as an application capable of recognising and identifying a set of known paintings was successfully developed for a mobile device. Currently museums and art galleries still rely on brochures, tour staff, and tape recordings to deliver information about the paintings hanging on their walls. The application proposed in this report aims alleviate the reliance on such antiquated ways to draw information from one's surroundings using the smartphone devices that most of us carry around with us now on a day-to-day basis.

This project aimed to develop an Android application capable for recognising and identifying paintings in an art gallery or museum setting. While many mobile computer vision applications related to object detection exist, many utilise the mobile device as simply a node for taking in input, sending it elsewhere for processing, and receiving the output result for displaying. For this project the entirety of the processing and computation takes place on the mobile device itself and while not perfect in terms of efficiency, performance, or accuracy shows that such an application may be viable as a dedicated application on the mobile platform if developed for accordingly.

The limitations of this project mainly lay in the technical aspects of the main algorithm developed in terms of accuracy and performance. Given the timeframe of the project some work was left incomplete which would have addressed some of these issues especially in regards to performance where serialisation of the data and further pre-processing of the data set would have made a huge impact on the speed in which extracted paintings could be matched to a reference painting. In terms of accuracy, further steps could be taken before utilising SIFT to match paintings based on colour.

Despite these limitations the results produced by the final application at the end of this project were promising enough to suggest that such an application may be viable on a mobile platform without relying on any backend system to do the background image-processing and computer vision related computations.

Bibliography

1. Reinhard Klette. Concise Computer Vision. Springer, 2014
2. Rick Penwarden. The Rise of the Smartphone, 2014. https://fluidsurveys.com/wp-content/uploads/2014/11/The-Rise-Of-The-Smartphone.pdf
3. Why Can't We Take Pictures In Art Museums? Art News, 2013. http://www.artnews.com/2013/05/13/photography-in-art-museums/
4. Expensive Journals Drive Academics to Break Copyright Law. NPR, 2016. http://www.npr.org/2016/02/20/467468361/expensive-journals-drive-academics-to-break-copyright-law
5. Vincent Gire, Sharareh Noorbaloochi. Painting Recognition using Camera-phone Images.
6. Theophile Dalens, Josef Sivic, Ivan Laptev. Painting Recognition From Wearable Cameras.
7. Niki Martinel, Christian Micheloni, and Gian Luca Foresti. Robust Painting Recognition and Registration for Mobile Augmented Reality.
8. Mark S. Nixon and Alberto S. Aguado. Feature Extraction and Image Processing. Academic Press, 2008, p. 88.
9. Cornell. Computer Vision: Filtering, 2013 http://www.cs.cornell.edu/courses/cs6670/2011sp/lectures/lec02_filter.pdf
10. S. Seitz - CSE 455, Washington edu
11. OpenCV Documentation. Canny Goals. http://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/canny_detector/canny_detector.html
12. OpenCV Documentation. Edge Theory. http://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/sobel_derivatives/sobel_derivatives.html
13. Kenneth Dawson-Howe. A Practical Introduction to Computer Vision with OpenCV, 2014.
14. OpenCV Documentation. SIFT. http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_feature2d/py_sift_intro/py_sift_intro.html
15. OpenCV Documentation. Contour Hierarchy. http://docs.opencv.org/3.1.0/d9/d8b/tutorial_py_contours_hierarchy.html#gsc.tab=0
16. OpenCV Documentation. Contours. http://docs.opencv.org/3.1.0/d4/d73/tutorial_py_contours_begin.html#gsc.tab=0

17. OpenCV Documentation. kNN. http://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_ml/py_knn/py_knn_understanding/py_knn_understanding.html
18. John Canny. A Computational Approach to Edge Detection, 1986.
19. David Lowe. Distinctive Image Features from Scale-Invariant Key-points, 2004.
20. Stanford Mobile Visual Search Dataset: Museum Paintings. http://web.cs.wpi.edu/~claypool/mmsys-dataset/2011/stanford/mvs_images/museum_paintings.html
21. Lei Xu. A New Curve Detection Method: Randomised Hough Transform, 1990.

Appendix

Demo

A demo of the final application can be found at the following URL
https://www.youtube.com/watch?v=SDjxd8QDUWc

Matching via colour using histograms

In order to match the paintings, a solution was explored whereby RGB histograms of the extracted painting and references would be compared to find a match.

Each of the three channels (red, green, and blue) would be represented by 255 bins for each intensity value.

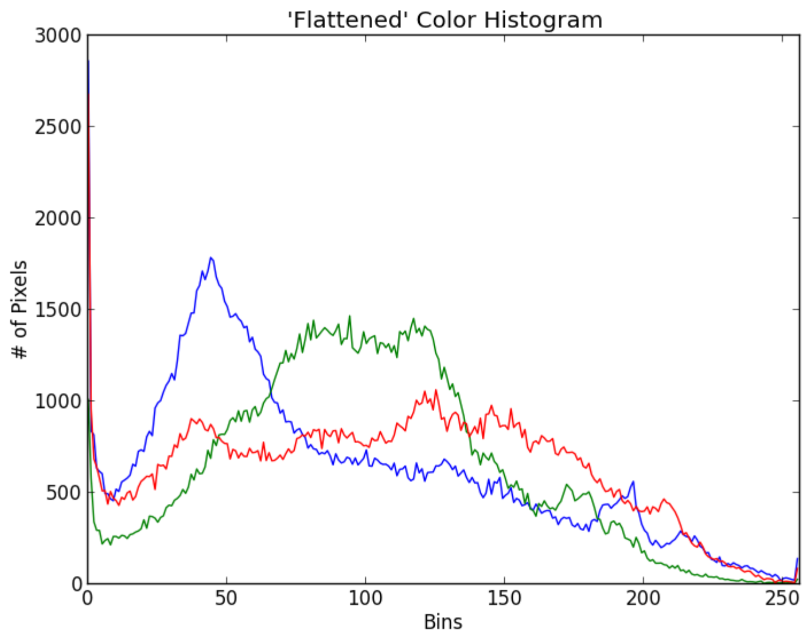


Figure 46 - Example of an RGB histogram for a painting

Once computed the histogram of the extracted painting would be compared to the pre-computed RGB colour histograms of the reference paintings.

In order to find the best match for the extracted painting, we would use the earth mover's distance (EMD) metric to judge which histogram was closest in comparison to the extracted painting's histogram.
Intuitively the easiest way to understand how EMD works is to consider multiple separate piles of sand. Imagine you are then assigned one pile and asked to find its closest match. The only moves you are allowed to make are to either move a single grain to or from your pile to another pile. So in order to find the closest match you would see which other sand pile you can make look exactly like yours in the fewest moves as this would indicate it was the sand pile that was most like yours and thus the best match. This what we essentially do when comparing the RGB histograms EMD to see which painting's histogram is likely the same as the extracted painting. We would expect the colour distributions to be quite similar given they are the same painting with the same colours, however we would also expect the distributions to be shifted one way or another due to factors such as illumination differences between the painting and its reference.