University of Glasgow | School of Computing Science

Honours Individual Project Dissertation

Autonomous Capture

Dan Cristian Cecoi
27th March 2019

Abstract

Every abstract follows a similar pattern. Motivate; set aims; describe work; explain results.

“XYZ is bad. This project investigated ABC to determine if it was better. ABC used XXX and YYY to implement ZZZ. This is particularly interesting as XXX and YYY have never been used together. It was found that ABC was 20% better than XYZ, though it caused rabies in half of subjects.”

Education Use Consent

Contents

1 | Introduction

1.1 Overview [temp]

Deep learning technologies are at the center of the current revolution in multimedia data analysis and artificial intelligence. Over the last years, convolutional neural networks have been shown to outperform previous state-of-the-art machine learning techniques for tasks such as object detection, human pose estimation and face detection. In the majority of cases, raw images are used to train deep learning models, which carries a big penalty in training time and efficiency, with weeks or months of training time in some cases.

A solution for this problem, as demonstrated by Ozimek (2017), uses a biologically inspired software retina for pre-procesing of images with proven reduced memory requirements and training times without compromising on training quality.

In this paper we present an approach to autonomously control a commercial pan-tilt-zoom security camera to capture a set of retinal transformed images used for training of Convolutional Neural Networks (CNNs).

1.2 Motivation [temp]

Biological vision systems are currently unrivaled by artificial visual systems for complex day to day tasks of an organism, such as navigating through the world, detecting dangers and identifying objects. Even the most simple tasks are still difficult for artificial systems. Due to this, biologically inspired vision systems seek to capture key aspects of the computational architecture of the brain and the sensor capabilities of the eye. Multidisciplinary inputs from other disciplines such as biology, neuroscience and cognitive science, will have a fundamental impact in the progress of computer vision. From the study of biological organisms, new types of computational paradigms can be achieved by providing a source of inspiration for new computational efficient and robust vision models.

With his biologically inspired software retina, Ozimek (2017) has demonstrated an approach for pre-processing images into a data efficient form based on the human vision system. Using the retinal transformed image data was shown to drastically decrease training times and the storage requirements while not compromising on the accuracy of the deep network models.

Furthermore, due to the decrease in prices of digital cameras and increase of overall quality, there is now a demand for using cheap equipment for both research and industrial computer vision applications without having to resort to using highly specialized and expensive camera systems.

Our goal is to synthesise these approaches into a real-time vision pipeline which is both time and space effective and is inexpensive to set up. An application is required which can use a cheap off-the-shelf security PTZ Camera for the purpose of identifying salient regions on an object, select and explore the most important points using a saccade targeting based gaze control mechanism and collect retinal transformed data-sets with those points as the center of attention.

This work extends on existing research on the biologically inspired software retina and work done by Zhou (2018) on a graphical user interface for manually controlling a PTZ camera.

1.3 Aims [temp]

There are three main objectives needed to be accomplished for the development of the autonomous capture mechanism:

1. The development of a non-proprietary Camera Controller component used for driving the camera.
2. The development of an Autonomous Capture component that independently detects salient points on objects within the field of view and direct the camera towards those points.
3. Storing the captured data-set and demonstrate its use by loading the data into the Data Management system and training an auto-encoder with it. This will be done in collaboration with Sean Tierney, a University of Glasgow MSci student working on the Data Management application.

2 Background

2.1 Mammalian Vision System

In the mammalian vision, light is perceived through a hemispherical layer of photoreceptor cells that convert photons into electrical signals through a series of chemical changes. The fovea, a central region of the retina where the highest density of the photosensitive cells are found, is responsible for the highest visual acuity, corresponding to 5 degrees of the visual field. Though the eye is capable of receiving visual information from a range of approximately 200 degrees, for most of that range the resolution is relatively poor due to decreasing density of the photoreceptor cells towards the peripheries of the retina.

Once the photoreceptor cells generate the electric signals, retinal ganglion cells (RGCs) transmit the visual signal to the primary visual cortex with majority of the RGCs being connected to multiple neighbouring photoreceptor cells. This grouping of photoreceptor cells connected to each ganglion cells is called the receptive field. The receptive fields are smallest in the fovea, where each ganglion cell is only connected to individual photoreceptive cells and increase in size the further from the fovea they are located.

By combining a small foveal region with visual attention mechanisms, biological organisms are saving on the brain's computational power and storage required to analyze a full visual scene. This has served as a motivation for numerous research into biologically inspired hardware and software retinas for computational and computer vision purposes. One such example on which this paper focuses is Balasuriya (2006) self-organised software retina. In his work, Balasuriya explores the generation, sampling function, feature extraction and gaze control mechanisms of the software retina.

2.2 Biologically Inspired Software Retina

Using a Self-Similar Neural Network, Balsuriya generated a receptive field tessellation with a near-uniform dense foveal region which smoothly transitions into a sparse periphery. Using this approach, the retina tessellation is created without generating local discontinuities or distortions. The positions of the Gaussian receptive fields in Balsuriya's software retina are governed by the node positions in the retina tessellation. The size of the receptive field, as is the case in the mammalian vision, scales with the distance from the center of the retina. Figure 2.2 shows an example of receptive field locations based on a tessellation generated using a Self-Similar Neural Network.

Once the input image is sampled using the generated receptive fields, the intensity values at each receptive field are stored in a 1-Dimensional *imagevector*. This vector is further used for the generation of the cortical and back-projected images.

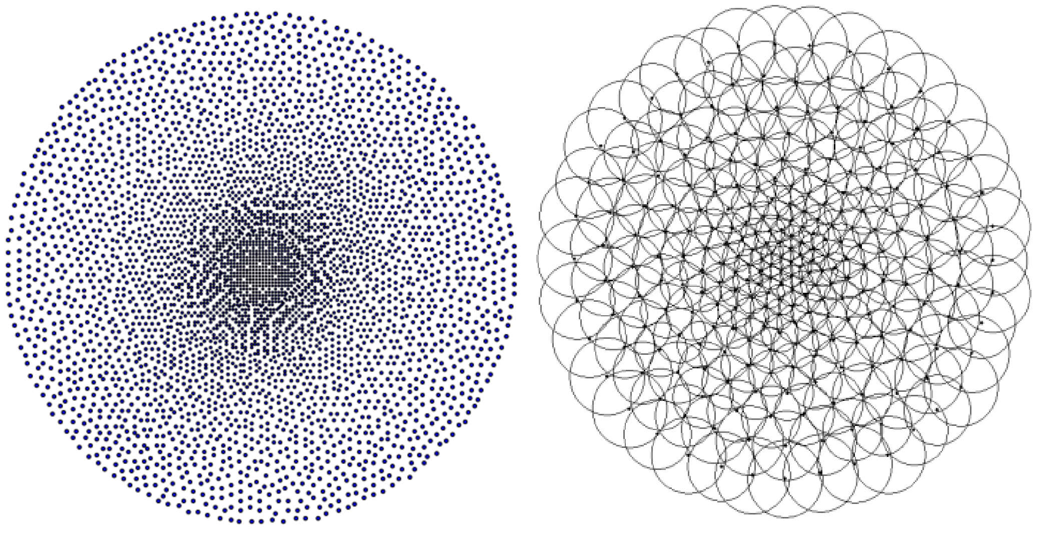

(a) Retina tesselation generated by Balasuriya *(b) Receptive fields generated from retina tesselation.*

Figure 2.1

2.2.1 Cortical Image Generation

Inspired by the primate visual cortex, the cortical images are data-efficient versions of the original image where the fovea is stretched out and the peripheries squashed. This is achieved by mapping the Gaussian receptive field centers to a "cortical space" and then project the associated imagevector intensities onto Gaussians centered on these locations.

The cortical space is based on the work of Schwartz (1977), which proposes a model for mapping the cartesian coordinates from image space to the log-polar coordinates (p, θ), with p being the distance from the origin and θ the angle about the origin.

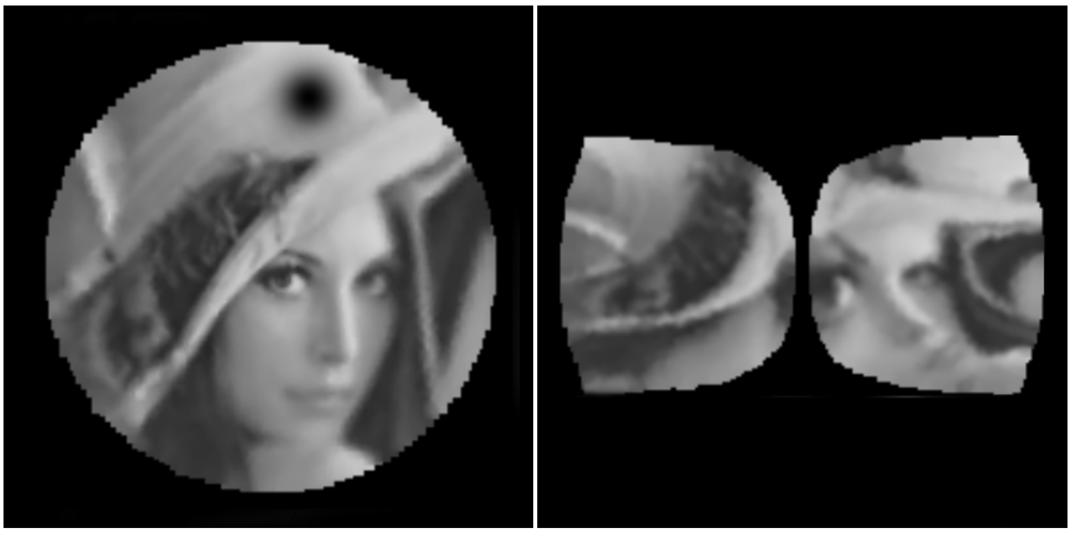
$$\begin{aligned} p &= log\sqrt{x^2 + y^2} \\ \theta &= \arctan(y/x) \end{aligned} \tag{2.1}$$

To remove the effects of excessive densities at the periphery and of the sparseness in the foveal region, the log operator has been removed and a further work of Schwartz (1980) has been adapted. In his revised approach, Schwartz adjusts the mapping with a parameter α while splitting the retina tesselation into two halves. Using equations 2.3, the cartesian coordinates in each half are mapped separately. This approach preserves local angles and mentains a uniform receptive field density without the introduction of noise.

$$\begin{aligned} X_{left} &= -\sqrt{(x-\alpha)^2 + y^2} \\ Y_{left} &= \arctan(y/(x-\alpha)) - sign(\arctan(y/(x-\alpha))) * \pi \\ X_{right} &= \sqrt{(x+\alpha)^2 + y^2} \\ Y_{right} &= \arctan(y/(x+\alpha)) \end{aligned} \tag{2.2}$$

2.2.2 Back-projected Image Generation

The back-projected image is used as an aid to visualize the responses of the space-variant sampling allowing to examine the sharpness and the artifacts generated by the retinal sampling. To create the back-projected image, every Gaussian receptive field is scaled by its corresponding intensity values in the imagevector and is back-projected onto the original image plane. The back-projected image has information only from the retinas field of view with the rest of the image being discarded. In figure 2.2a a high resolution center corresponding to the fovea and a lower resultion periphery can be observed.



(a) Back-projected image. *(b) Cortical image.*

Figure 2.2

2.3 Local Feature Detection

Image local feature extraction is a dimensionality reduction technique where the resultant information provides a low dimensional description of the image contents. Local image features can be defined as a specific pattern that is unique to its immediately close pixels. These patterns are generally associated with a change of an image property or several properties simultaneously such as intensity, colour or texture. Local features are generally points but can be corners or blobs (Tuytelaars and Mikolajczyk 2008). These points serve as anchors whose neighbourhoods are represented by computed feature vectors.

Local feature extraction is the most used low-level visual representation due to robustness to occlusion and clutter, their distinctiveness, repeatability and efficiency with achievable real-time performance. They can be used in applications such as image alignment, object recognition and motion tracking.

2.3.1 Scale Invariant Feature Transform

A wide variety of feature description and detections algorithms have been proposed in literature. One of the most robust and scale-invariant feature detectors with high repeatability is Lowe's *Scale Invariant Feature Transform* (SIFT). SIFT algorithm has four main steps:

1. Scale Space Extrema Detection
2. Key point Localization
3. Orientation Assignment
4. Description Generation

In order to identify interest points that are scale and orientation invariant, locations and scales that can be repeatably assigned under differing views of the same subject must be identified. To achieve this, the scale space is separated into octaves and in each octave the initial image is repeatedly convolved with Gaussians to produce a set of scale space images. To find local extrema, each level of the pyramid has its adjacent Gaussians subtracted to produce a *Difference of Gaussians* (DoG).

To accurately locate interest points, key point candidates are localised and refined by using a Taylor series expansion of the scale space to get a more accurate location of extrema. The intensity at these extrema is compared to a contrast threshold value, and if its lower the extrema is rejected. However, because the DoG function has a higher response along edges, rejecting only key points with low contrast is not sufficient and the edge responses need to be removed as well. This is achieved by computing a 2x2 Hessian matrix, H, at the location and scale of the keypoint, from which the principal curvature can be calculated. A poorly defined peak will have a large principal curvature across the edge but a small one in the perpendicular direction. The eigenvalues of H are proportional to the principal curvatures of the scale space function. Fortunately, it is not necessary to directly compute the eigenvalues as we are only interested in the ratio between the largest magnitude eigenvalue and the smaller magnitude eigenvalue. This ratio is compared with a threshold, r, discarding the keypoints whose ratio of principle curvatures is over that threshold (Lowe 2004).

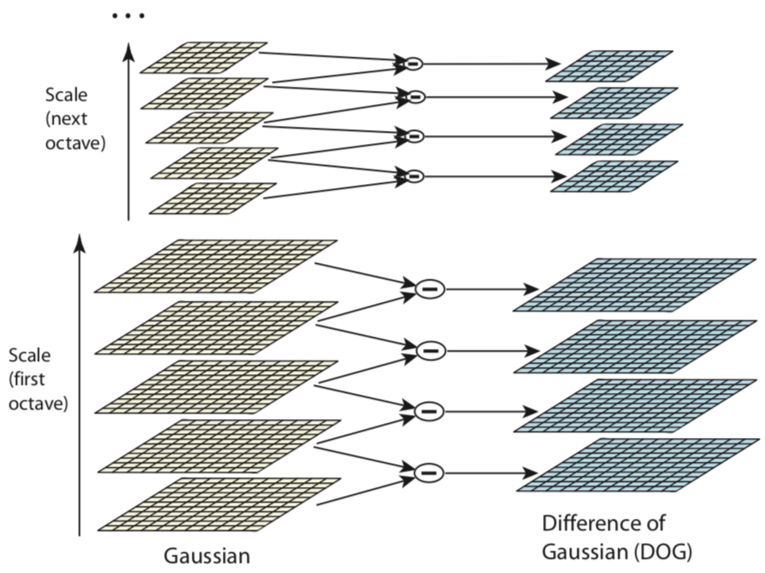


Figure 2.3: *Each octave of the scale space is repeatedly convolved with Gaussians to produce the scale space images on the left. To produce the difference-of-Gaussian shown on the right, adjacent Gaussian images are subtracted. After each octave, the Gaussian image is down-sampled by a factor of 2, and the process repeated (Lowe 2004).*

Orientation assignment is achieved by creating a orientation histogram of local gradient directions around the keypoint depending on the scale. The highest peak in the orientation histogram corresponds to the dominant direction of the key point.

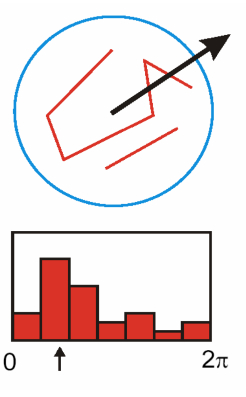


Figure 2.4: Orientation Assignment by local gradient direction. (Lowe 2004)

In the final step of the algorithm, the keypoint descriptor is created. A 16-by-16 neighborhood around the keypoint is taken and subdivided into 4-by-4 sub regions. A orientation histogram is created over these sub regions by evaluating the magnitude of gradients in eight directions. All the orientation histograms are combined together into a 128 dimensional SIFT point descriptor. In addition to this, the feature vector is normalized in order to achieve robustness against illumination changes.

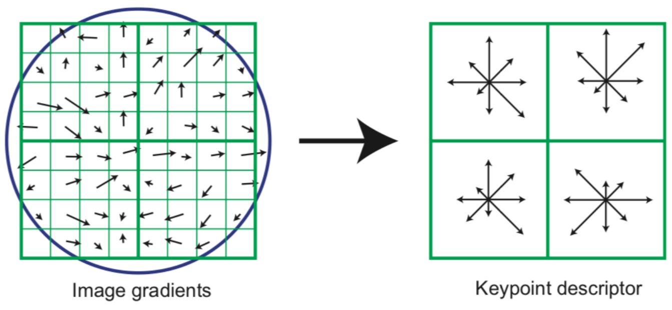


Figure 2.5: 2x2 descriptor array computed from an 8x8 set of samples (Lowe 2004).

2.3.2 Speeded Up Robust Features

In the previous sub-section, SIFT algorithm for key point detection and description was described. While it became the most popular algorithm to extract distinctive invariant features from images, it was comparatively slow and a faster method was needed.

In 2006, Bay and Tuytelaars presented the Speeded Up Robust Features (SURF) algorithm that was partially inspired from SIFT, with similar steps but with different implementation details.

To find points of interest, SURF uses a "Fast-Hessian" Detector which is based on the Hessian matrix due to its good performance in computation time and accuracy. Given a point p=(x,y) in an image, the Hessian Matrix $H(p, \sigma)$ at point p and scale σ, is:

$$H(p,\sigma) = \begin{bmatrix} L_{xx}(p,\sigma) & L_{xy}(p,\sigma) \\ L_{yx}(p,\sigma) & L_{yy}(p,\sigma) \end{bmatrix} \tag{2.3}$$

Where $L_{xx}(p,\sigma)$ is the convolution of the Gaussian second order derivative with the image in point x.

In SIFT, the approximation of Gaussian smoothing is achieved by using cascaded filters to detect scale-invariant characteristic points where the DoG is calculated on rescaled images. SURF pushes the approximation of the Laplacian of Gaussian even further and approximates it with box filters. These approximate second order Gaussian derivatives, and can be evaluated very efficiently using integral images, regardless of size, with the added benefit of removing the effects of aliasing when the images are sub-sampled (Bay et al. 2008).

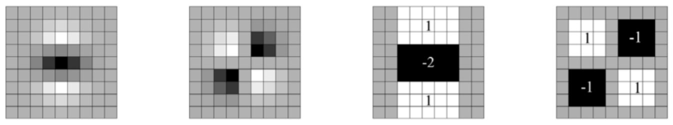


Figure 2.6: *The approximation of Gaussian second order partial derivatives using box filters (Bay et al. 2008).*

Interest points at different scales are found by applying box filters of different sizes. The scale space is analyzed by up-scaling the filter size rather than iteratively reducing the image size. The following layers are obtained by filtering the image with gradually bigger masks, hence, for each new octave, the filter size increase is doubled.

For the purpose of descriptor computation, the orientation of the point of interest must be determined. This is achieved by computing the Haar wavelet responses in horizontal and vertical directions around the point of interest. The dominant orientation is estimated by calculating a vector from the summation of all responses within a sliding orientation window covering a 60 degrees angle. Finally, the longest vector is selected which lends its orientation to the interest point.

The descriptor is based on the sum of Haar wavelet responses. The region around the point is described by extracting a square region around the point of interest. The size of the region is directly proportional to the scale and is oriented along the orientation computed in the previous step. This region is split into smaller 4x4 square sub-regions and for each sub-region the Haar wavelet response is extracted. To offer more robustness for deformations, noise and translation the responses are weighted with a Gaussian. Finally, the responses are summed up over each sub-region to form a first set of entries to the feature vector.

Bay et. all estimate that SURF outperforms SIFT by 4 times on average in computation time while not sacrificing in quality and robustness.

2.4 Local Feature Matching

2.4.1 Matching strategies

Matching descriptors from one set with the descriptors in a second set is usually performed by using two approaches: Brute Force Matching or applying a indexing structure. Brute Force

Matcher is the simplest of the two. It functions by matching a descriptor from the first set with all the descriptors from the second set, returning the closest one based on the Euclidian distance (Lowe 2004). While this is not efficient, it is guaranteed to return the best matches of the two sets.

For the purpose of achieving a better matching performance, an indexing structure is devised. One of the most widely used indexing structures are the k-d trees, which are a class of multi-dimensional search trees. This matching algorithm is much faster for larger data sets, however, it is not guaranteed to return the best match, as it only returns the approximate nearest neighbours.

Outlier rejection from the feature space is accomplished by comparing the distance value of the closest match to the distance value of the second-closest match. In his seminal paper, Lowe suggests to reject all the matches with a distance ratio greater than 0.8, which was the approach used in this project.

In order to further eliminate the remaining error matches, a robust estimation method introduced by Fischler and Bolles (1981) called Random Sample Consensus (*RANSAC*) is frequently used to to handle mapping features in the presence of outliers.

2.4.2 Homography

Inlier correspondences between two images are found by computing the planar homography of the transformation. A homography, also known as plane projective transformation, is a 3 by 3 matrix H that relates the transformation between corresponding image points as:

$$x' = Hx \tag{2.4}$$

Two images are related by a homography if and only if the image points correspond to the same planar surface in the real world and the images were acquired by rotating the camera about its optical axis.

To compute the homography between two image pairs, RANSAC robust estimation is applied. For each iteration of the RANSAC algorithm:

1. A random sample of 4 matches is selected and the homography H is computed.
2. The geometric image distance error for each match is calculated.
3. The number of inliers from the number of matches for which the distance error is less than a threshold is computed.
4. The homography with the largest number of inliers is selected.
5. The least-squares homography estimate using all of the inliers is re-computed (Torr and Zisserman 2000).

2.5 Gaze Control

One of the most important skills required for both biological organisms and artificial machines is the ability to direct the vision system towards a target. For humans, eye movements are split into mechanisms to either stabilize gaze or redirect gaze. The redirecting gaze mechanisms which are responsible for orienting the eyes to point to new locations are split into two classes: saccade movement and smooth pursuit movement. Saccades are rapid eye movements that allow us to quickly scan a visual scene and are characterised by quick, step-like rotations between fixation points, where the image of the visual target located at the perifery of the retina is moved to the center. Following each saccade, the eyes remain stationary at their new positions enabling the subsequent high-acuity detailed visual analysis at that location. The destinations of each saccade movement are selected by cognitive brain processes, with vision being dependent upon the information gathered during the pauses between saccades. Contrary to the rapid, step-like saccade movement, smooth pursuit eye movements slowly rotate the eyes in a linear fashion in order to keep a moving stimulus centered on the fovea. Because this work is focused on a stationary visual scene, where the only movement is done by the PTZ Camera, the gaze control will only be based on the saccade movement.

Haitham Fattah (2008) used saccade targeting as a means for gaze control, in which the attention system selects new saccade movements based on computed SIFT local features in the view of the cameras. To stop the camera from returning to locations that have already been attended, an inhibition of return (IOR) mechanism was implemented by maintaining a list of already visited points. Similarly, Ozimek (2017) selects fixation points for the software retina by populating a saliency map with SIFT features while maintaining an inhibition of return map where each fixation point is represented by amplified Gaussian spanning the retinal foveal region. By subtracting the IOR map from the saliency map, points that were previously visited are discarded. These two approaches are an inspiration for the gaze control used in this paper.

3 Materials and Methods

In this chapter we give an overview of the main hardware and the software components we have used for this project.

3.1 Pan-Tilt-Zoom Camera

The camera used in this project is a network PTZ Camera manufactured by LEFTEK which is capable of remote directional and zoom controls. Even though cameras of this type were designed for surveillance applications they are also suitable for collecting images for the purpose of training Neural Networks. The benefits of using this camera arise from it's relative cheap cost while providing a high resolution sensor and a wide angle lens both of which are necessary for capturing a high amount of details from a wide angle of view. In addition, it's PTZ capabilities provide the possibility for capturing images from different angles and distances.

Figure 3.1: *LEFTEK pan-tilt-zoom camera used in this project*

This camera uses a number of protocols for controlling the camera and for video streaming, the most important of them being: Hyper Text Transfer Protocol (HTTP), Real Time Streaming Protocol (RTSP) and Open Network Video Interface Forum (ONVIF).

3.2 OpenCV

OpenCV (Open Source Computer Vision) is a cross-platform library aimed at real-time computer vision, image processing and machine learning written in C++. Even though OpenCV is mostly used in applications developed with C++, this project uses the Python OpenCV library that exposes the C++ functions through Python bindings.

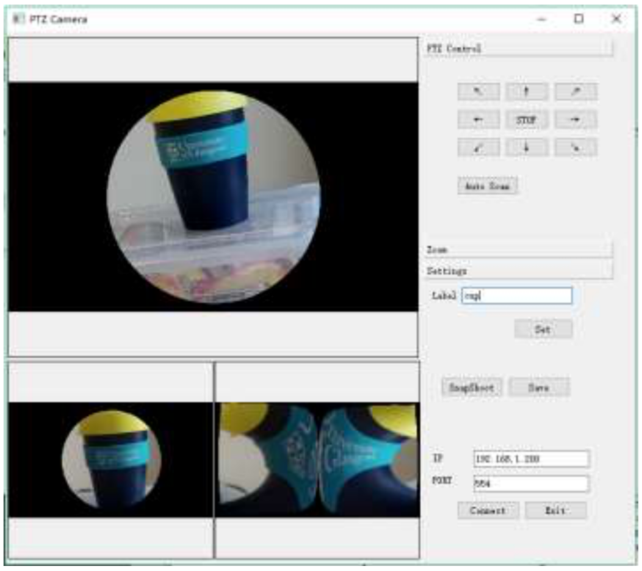
Beside basic Image or video Input/Output processing and displaying functionality, it has a set of built in complex algorithms for object and feature detection, Machine Learning, Computational Photography and others. In this paper, we are mostly focused on the features2D module that provides methods for 2D image feature and descriptor extractors such as SIFT, SURF and ORB.

Though there are other alternatives for image processing, none have the amount of built-in functionality OpenCV provides. Furthermore, because the ORB feature detection algorithm was developed by "OpenCV Labs", it solidified our decision of it being the framework of choice.

3.3 Graphical User Interface

This project has inherited the Graphical User Interface (GUI) developed by Zhou (2018). His work focused on integrating Ozimek's software retina into a tailored manual capture control interface capable of controlling the same camera used in this project. This GUI provides functionality for connecting to the camera and displaying a live view of the video stream. Additionally, it provides PTZ Controls which are implemented using the manufacturers proprietary HTML protocol. Finally, in terms of the software retina integration, it allows for the creation and saving of retinal transformed images and displaying the cortical and back-projected images.

In this project we seek to use this GUI as a baseline into which the Autonomous Capture component will be integrated, implementing the necessary modifications necessary to fulfill this project's requirements.



Label cup1

IP 192.168.1.200

PORT 554

Figure 3.2: *Graphical User Interface developed by Zhou (2018)*

4 Implementation

4.1 ONVIF Protocol

One of the key requirements concerning the Controller component was for it to be implemented in a non-proprietary protocol. Because it is supported natively by the PTZ Camera, and because it is by far the most popular open standard for the interface of physical IP-based security products, *Open Network Video Interface Forum* (ONVIF) was chosen as the protocol in which this component will be implemented. ONVIF defines a standard for how IP Camera products within the surveillance industry can communicate with each other. By implementing ONVIF, the same instructions such as initiating a video stream or moving the camera can be used for a big number of devices across different manufacturers. Therefore, if in the future there is a decision to replace the camera with a different model that supports the ONVIF protocol, it wont be necessary to rewrite the base implementation of the Controller.

On the most basic level, in the ONVIF protocol, the functions exposed to the client are defined in Web Service Description Language (WSDL) files corresponding to each ONVIF service and the commands are sent via Simple Object Access Protocol (SOAP) requests. A developer can choose to implement the functions based on the WSDL files and by following the ONVIF Core Specification. In our application we make use of ***onvif-python*** library that offers an already implemented Access Point Interface(API) for these functions. Particularly, we utilise the ONVIF PTZ Service which defines the web service interface for configuration and operation pan-tilt-zoom controllers.

One of the big limitations of the ONVIF protocol is that for a camera to be considered ONVIF compliant only the "core" specification must be implemented by the manufacturer. In the Move Operations specification there are defined 3 different PTZ move operations: AbsoluteMove, RelativeMove and ContinuousMove. In AbsoluteMove the position argument of this command specifies the absolute position to which the PTZ unit moves. In RelativeMove the translation argument specifies the difference from the current position to the position to which the PTZ device is instructed to move. In ContinuousMove, the velocity argument of this command specifies a signed speed value for the Pan, Tilt and Zoom. Even though AbsoluteMove and RelativeMove offer an increased granularity and sensitivity for the move operations, ContinousMove was the only one to be implemented by the manufacturer of the PTZ Camera used in this project.

In ContinousMove operation the PTZ command is executed continuously until a stop command is received by the camera. While this is perfectly adequate for applications such as controlling the camera from a joystick or by pressing buttons, it is not very suitable for the level of control required in this project. In order to move the camera a certain amount of degrees or pixels, the movement of the camera must be timed.

In the prototyping phase, the velocity of the camera for Pan and Tilt was estimated by timing how much time it takes to complete a full rotation in each dimension. However, it soon became clear that there are significant errors caused by the network lag, the time the camera takes to process the commands and the inaccuracies caused by the PTZ motors not having a consistent velocity in each direction.

To lessen the effect of these deficiencies, a predictive model that takes these variables into account needed to be developed.

4.2 Controller Model

With the aim of minimising the effect of the limitation described in the previous sub-section, a Linear Regression model has been trained that will estimate a move time prediction taking into consideration the network delay, the signal lag and the physical inaccuracies of the camera. During the data gathering process it also soon became clear that the camera does not only move differently based on it being a pan or tilt movement, it also has a different behavior when it moves in different horizontal and vertical directions. Due to this observation, it was decided to train 4 different models depending on the movement that is required: Pan-Left, Pan-Right, Tilt-Up, Tilt-Down.

4.2.1 Data Gathering

In order to train the Linear Regression model, it was necessary to gather data of the camera pixel displacement per time interval. To achieve this, PTZ Move requests were sent to the camera with the time interval increasing with steps of 10 milliseconds. Data was collected for each of 4 pan-tilt directions, with an average of 1100 data points collected for the tilt models. Owing to a bigger range of movement in the horizontal direction, for each of the pan models approximately 2600 points were collected.

To calculate the pixel displacement between two image pairs, a process based on a feature detector needed to be used. While the quality of the feature detection and matching was a vital characteristic in the search of a suitable algorithm, because the pixel displacement for over 7400 image pairs needed to be computed, the computational requirements also needed to be considered. SURF was selected over SIFT as the feature detector algorithm of choice for this task because of its decreased computation time with a similar quality of matches as SIFT.

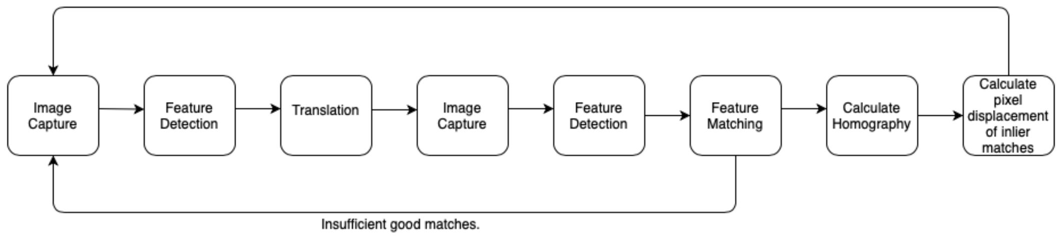


Figure 4.1: *Data Gathering Process*

In the first part of the data gathering process, an image of an object is taken, and its features are detected using SURF. After the first set of features is computed, the camera translates for a specific amount of time and a second image is taken from which a second set of features is computed. To match the features between the two images, a choice needed to be made between the Brute Force Matcher (BFMatcher) and Fast Library for Approximate Nearest Neighbors (FLANN) based K-d tree algorithm. While FLANN has a lower computation time, the quality of the matches suffer as it is not guaranteed to produce the best matches. Because the quality of the matches was more important in this task than the increase in speed, BFMatcher was chosen as the matching algorithm.

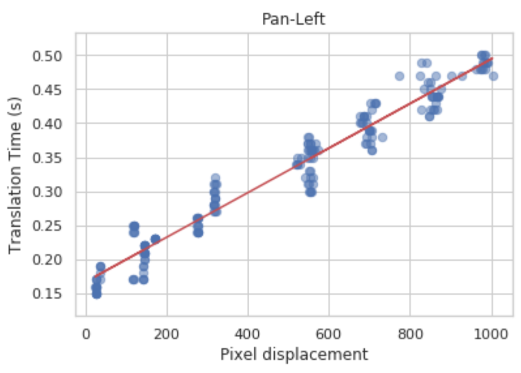
Once the potential matches were computed, the error matches were removed in a two part process. Firstly, a part of the error matches were removed by applying Lowe's ratio test described in the Background section. In the second part of the process, the homography of the transformation between the two images is computed using RANSAC and the outlier matches are rejected.

Finally, the pixel displacement of every inlier match is computed and the median value is calculated in order to minimise the influence of any remaining outliers that were not rejected previously.

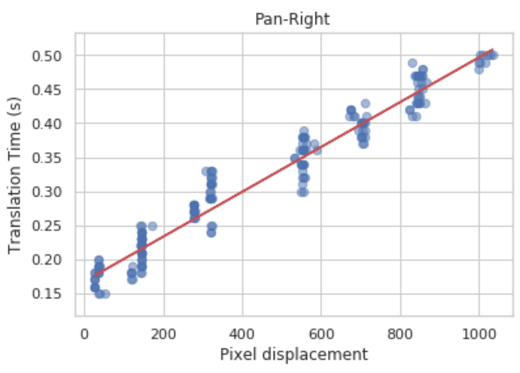
4.2.2 Model Training

After analysing the plotting the collected data, it became apparent that there is a linear relationship between the camera translation time and the pixel displacement. Therefore, it was decided to train simple linear models for the Camera Controller.

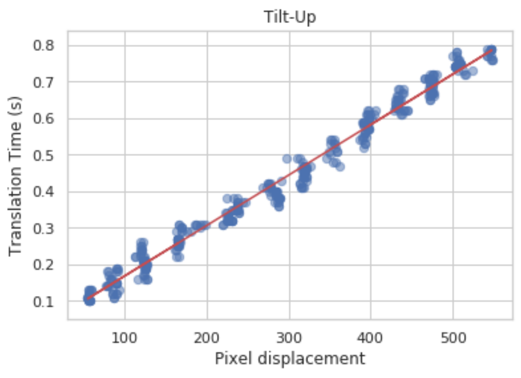
Firstly, a suitable choice for a framework capable of training a machine learning predictive model needed to be made. *Scikit-learn* is a very powerful, open source library based on SciPy, NumPy, and matplotlib. While there are numerous others Machine Learning frameworks, such as TensorFlow, Keras, PyTorch, which are more suitable for more complex Machine Learning models than a linear regression, Scikit-learn was chosen for the familiarity, simplicity of use and the great built in support for different kinds of generalized linear models.



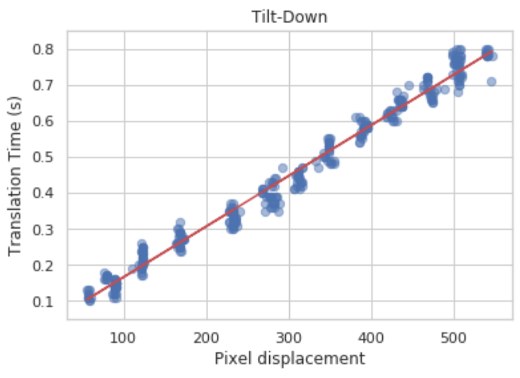
(a) Pan Left Model



(b) Pan Right Model



(c) Tilt Up Model



(d) Tilt Down Model

Scikit-learn offers built in support for a big number of Linear Regression models, the most important of them being: Ordinary Least Squares, Lasso and Elastic Net. The later two are powerful techniques used for creating models in the presence of a very large number of features. The models apply regularisation techniques in order to penalise the magnitude of the less important features in the data set. Because the data used for training the models have only a feature per data point, the use of Lasso or Elastic Net was deemed unecessary and Ordinary Least Squares was used for training.

Least Squares regression is the most common method for fitting a regression line. The underlying relationship between a dependent variable y_i and an explanatory variable x_i involving the error term ϵ_i can be described by:

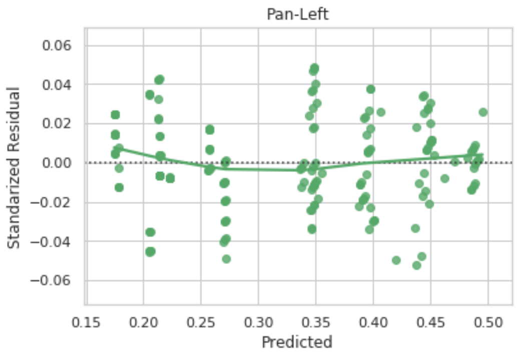
$$y = \alpha + \beta x_i + \epsilon_i \tag{4.1}$$

This method calculates the best-fitting line for the observed data by finding estimated values $\hat{\alpha}$ and $\hat{\beta}$ which minimise the sum of the squares of the vertical deviations from each data point to the line.

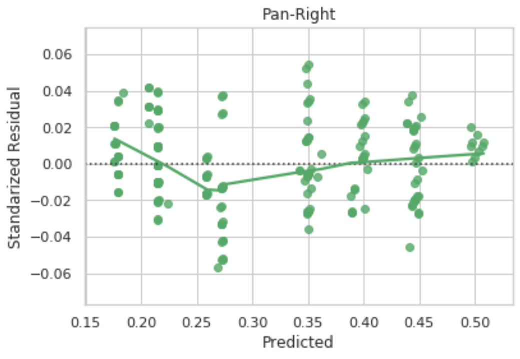
As part of the data preparation step, the outliers in the data were identified and removed by calculating the interquartile range (IQR) scores. IQR is is a measure of statistical dispersion, being equal to the difference between the upper and lower quartiles. While both variance and standard deviation are a measure of dispersion, IQR is much more robust to outliers. The data points that were above 1.5 * IQR the 3rd quartile and below the 1st quartile were removed from the data.

Based on the collected data, there was no translation between the time intervals of 0.0-0.15 ms for Pan and 0.0-0.1 ms for Tilt, therefore the data points corresponding to those time intervals were removed from the training set, resulting in better fitted models with a variance score of 0.95 for Pan models and 0.98 for Tilt models.

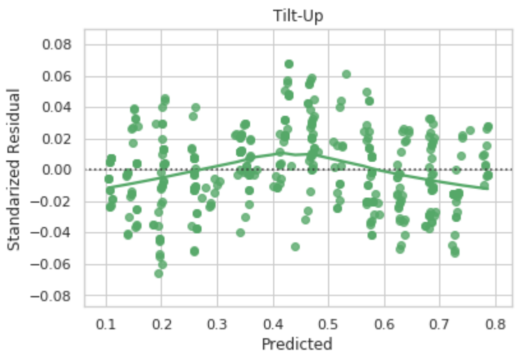
The residual plots show that the difference between the predicted and real values are clustering around the lower y-axis digits. The data is also distributed on both sides of the y-axis, therefore the models don't tend to predict neither too low nor too high on average. However, one of the key requirements of a good residual plot is that there are no clear patterns on it. There is a clear quantisation of data observed in both the residual plots and in the Translation per Pixel displacement graphs.



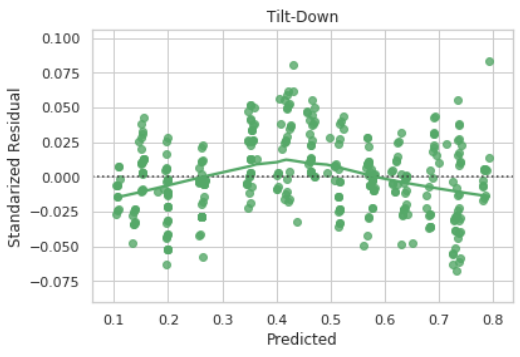
(a) Pan Left Model Residual



(b) Pan Right Model Residual



(c) Tilt Up Model Residual



(d) Tilt Down Model

To remove the possibility of errors in the matching process and the homography computation to be the underlying reasons for this grouping of data points, for a series of image pairs the pixel displacement was manually determined and compared with the pixel displacement computed using the data gathering process described above. The results were consistent, having an average error of less than 10 pixels. Hence, no deficiency in the data gathering process has been found and the physical characteristics of the camera were determined to be the reason for this phenomenon.

Based on the collected data and the models, for a 1920 by 1080 pixels image, the smallest translation for pan is 37 pixels and 57 pixels for tilt.

4.2.3 Controller Architecture

After training a scikit-learn model, it is necessary to save the model for future use without having to retrain. This was achieved using Python's built-in persistence model called Pickle. Pickle is used for serializing and de-serializing Python object structures by converting Python object hierarchies into byte streams that can be saved to files.

To direct the camera to a point on an object, the pixel coordinate on the image must be determined. Images in NumPy are multidimensional arrays, where each pixel value is stored as an element of the matrix that can be accessed by using NumPy's row-major-order indexing. Once the pixel coordinate has been selected, the Controller computes the direction and pixel distance of the point from image center. Depending on the direction, the suitable linear model is selected. The pixel displacement value is used as an input for that model with the output being the translation time prediction required for the camera to move in order to center on that specific point.

As mentioned above, the camera moves continuously from the moment a PTZ Command is received until the moment a Stop command is sent. This interval between the two commands is implemented using the built-in Python sleep() function used for pausing the execution of the program for a specific amount of time. Albeit this function is only accurate in the order of milliseconds and can be affected by the scheduling of other activities in the system and the implementation of the underlying Operating System's sleep function, it is perfectly acceptable for driving the camera with a similar expected performance across different Operating Systems.

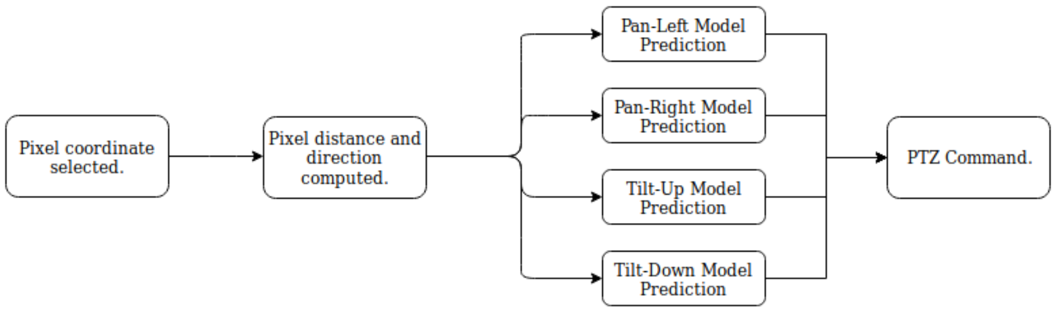

Pixel coordinate selected.
Pixel distance and direction computed.
Pan-Left Model Prediction
Pan-Right Model Prediction
Tilt-Up Model Prediction
Tilt-Down Model Prediction
PTZ Command.

Figure 4.4: *Camera Controller*

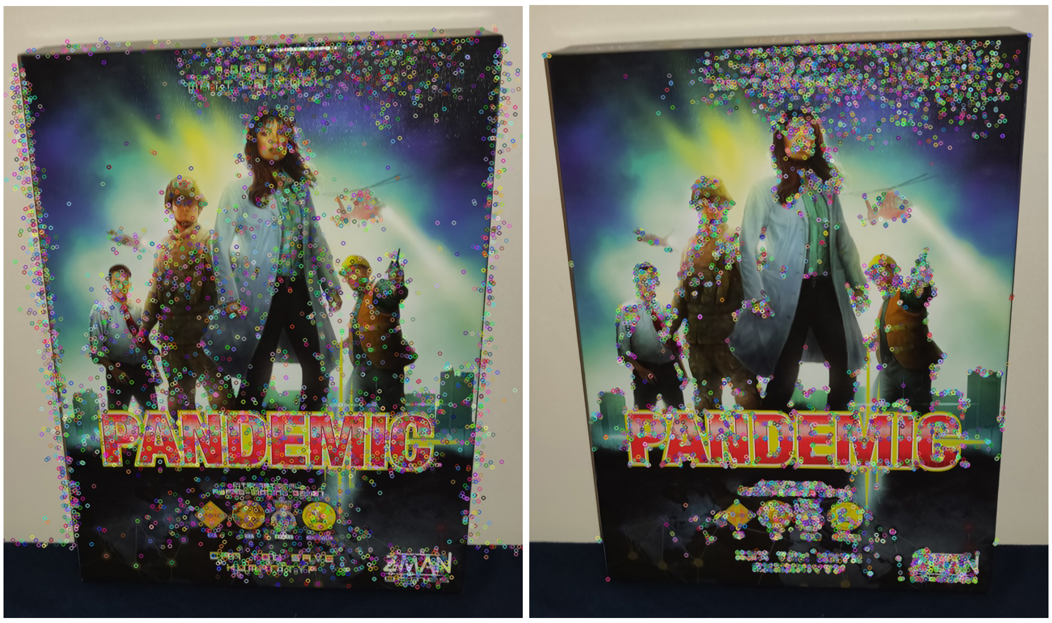
For the purpose of increasing the accuracy of the Camera movement, a simple retargeting algorithm was implemented. After a key point is selected and the camera translates to that point, local features are detected in the image after translation and the new position of the point is found. If the point is not found within the acceptable error margin from the image center, small incremental adjustments are performed using the smallest movement the camera is capable of until the point is within the acceptable distance.

4.3 Gaze Control

The gaze control mechanism in this work was inspired by the pre-attentive processing of visual information in the mammalian brain. In this subconscious process, an analysis of the visual information is performed on the entire visual field, detecting basic features of objects such as colors, contrasts and contours. After all this information is collected, the brain filters the information, selecting the information that has the highest saliency.

For the purpose of detecting salient features on objects two different feature detectors were considered: SURF and ORB. In the initial stages of the development, OpenCV implementation of SURF was used for feature detection. Despite the fact that SURF is a much faster feature detector than SIFT, it proved to be too slow for real time feature detection. To make the final choice, the performance of SURF and ORB was compared. For the same image, ORB detected 11589 features with an average computation time of 0.01747 seconds. SURF, on the other hand, detected 5780 features in 0.15748 seconds, which corresponds to approximately 10 times decrease in speed for half the amount of features detected. Because computational time was of an essence in this application, ORB was selected for the final implementation.

Once the local features of an object are computed, the key points are sorted and the most salient points are selected. Based on the work of Wong (2017), there are two factors that affect the choice of the focal point. The first factor is the intrinsic ORB keypoint response value which defines how strong the keypoint is. The second factor is the distance on the horizontal axis of the keypoint from the center of the image. Assigning a higher priority to keypoints further away from the image center allows for the exploration of other ares of the image without the same region of the object being explored continuously. The horizontal distance is normalised between 0 and 1 by dividing it with half the image width.



(a) *SURF features.* *(b)* *ORB features.*

Figure 4.5: *Comparison of features detected by SURF and ORB*

Using equation 4.2, the new key-point rank, *kprank* is calculated assigning higher weight to strong key-points further away from origin.

$$kp_{rank} = \frac{(kp_{loc} - origin_{loc})}{(cort_{width}/2)} \times kp_{response} \tag{4.2}$$

To avoid visiting the same regions of the object, an Inhibition of Return mechanism is implemented by saving in memory the key-points that were previously selected.

Once the *kprank* is calculated for every key-point, they are sorted by their new rank. Before a new focal point is selected, the list of new key-points is matched with the IOR list and the matched key-points are discarded.

[!!!TALK ABOUT CHOICE OF FLANN vs BFMATCHER!!!!]

Very frequently there are small regions with tightly packed key-points, which is also observed in figure 4.5. To avoid visiting key-points which are in the immediate proximity of a previously selected key-point, a circular mask around the coordinates of the key-point is applied, and the key-points withing the mask are detected and added to the IOR list.

Once the already visited focal points are discarded, the remaining key-point with the highest value is selected.

4.4 Autonomous Capture

4.5 GUI Modifications

5 Evaluation and Analysis

5.1 Gaze Control Evaluation

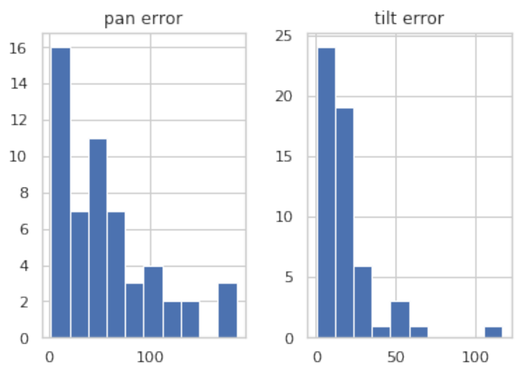
5.1.1 Performance

5.1.2 Quality

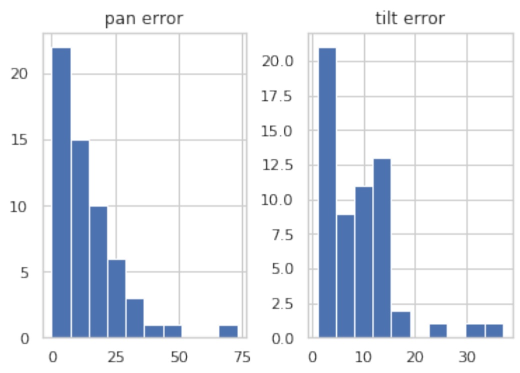
5.2 Controller Model Evaluation

Once the models have been trained, the real-use case of the Camera Controller and the effectiveness of the Linear Regression models was tested. In order to get a good sample of key points with a good spread over the image, random key points were selected from the set. After the Camera movement finishes, the matching key point post translation is determined and the new distances are calculated.

Using the retargeting algorithm improved the quality of translation drastically with a mean error of 9 pixels for tilt and 14 pixels for pan compared to 17 and 55 pixels respectively. This corresponds to approximately 1% error.



(a) Error without retargeting.



(b) Error with retargeting.

5.3 Overall Evaluation

6 Conclusion

6 Bibliography

S. Balasuriya. A computational model of space-variant vision based on a self-organised artificial retina tessellation. 2006.

H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359, June 2008. ISSN 1077-3142. doi: 10.1016/j.cviu.2007.09.014. URL `http://dx.doi.org/10.1016/j.cviu.2007.09.014`.

M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981. ISSN 0001-0782. doi: 10.1145/358669.358692. URL `http://doi.acm.org/10.1145/358669.358692`.

J. P. S. Haitham Fattah, Gerardo Aragon-Camarasa. Towards binocular active vision in a robot head. *Towards Autonomous Robotic Systems 2008*, pages 25–32, 2008. URL `http://cas.ee.ic.ac.uk/people/cmwitk/TAROS_2008_proceedings.pdf`.

D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60 (2):91–110, Nov. 2004. ISSN 0920-5691. doi: 10.1023/B:VISI.0000029664.99615.94. URL `https://doi.org/10.1023/B:VISI.0000029664.99615.94`.

P. Ozimek. Integrating a non-uniformly sampled software retina with a deep cnn model. 2017.

E. L. Schwartz. Spatial mapping in the primate sensory projection: Analytic structure and relevance to perception. *Biological Cybernetics*, 25(4):181–194, Dec 1977. ISSN 1432-0770. doi: 10.1007/BF01885636. URL `https://doi.org/10.1007/BF01885636`.

E. L. Schwartz. Computational anatomy and functional architecture of striate cortex: A spatial mapping approach to perceptual coding. *Vision Research*, 20(8):645 – 669, 1980. ISSN 0042-6989. doi: https://doi.org/10.1016/0042-6989(80)90090-5. URL `http://www.sciencedirect.com/science/article/pii/0042698980900905`.

P. H. S. Torr and A. Zisserman. Feature based methods for structure and motion estimation. In *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice*, ICCV '99, pages 278–294, London, UK, UK, 2000. Springer-Verlag. ISBN 3-540-67973-1. URL `http://dl.acm.org/citation.cfm?id=646271.685642`.

T. Tuytelaars and K. Mikolajczyk. Local invariant feature detectors: A survey. *Found. Trends. Comput. Graph. Vis.*, 3(3):177–280, July 2008. ISSN 1572-2740. doi: 10.1561/0600000017. URL `http://dx.doi.org/10.1561/0600000017`.

R. Wong. A smartphone software retina. 2017.

J. Zhou. Manual capture control interface for retina vision. 2018.