LASER INTERFEROMETER GRAVITATIONAL WAVE OBSERVATORY
- LIGO -
CALIFORNIA INSTITUTE OF TECHNOLOGY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Technical Note	**LIGO-T1900280–v2**	2020/01/31

Laser beam position tracking for LIGO interferometers

Milind Kumar V
Mentors: Rana Adhikari, Gautam Venugopalan, Koji Arai

California Institute of Technology
LIGO Project, MS 18-34
Pasadena, CA 91125
Phone (626) 395-2129
Fax (626) 304-9834
E-mail: info@ligo.caltech.edu

Massachusetts Institute of Technology
LIGO Project, Room NW22-295
Cambridge, MA 02139
Phone (617) 253-4824
Fax (617) 253-7014
E-mail: info@ligo.mit.edu

LIGO Hanford Observatory
Route 10, Mile Marker 2
Richland, WA 99352
Phone (509) 372-8106
Fax (509) 372-8137
E-mail: info@ligo.caltech.edu

LIGO Livingston Observatory
19100 LIGO Lane
Livingston, LA 70754
Phone (225) 686-3100
Fax (225) 686-7189
E-mail: info@ligo.caltech.edu

http://www.ligo.caltech.edu/

Acknowledgments

I am extremely grateful to Prof. Rana Adhikari for giving me the opportunity to work at the 40m Prototype Interferometer Laboratory and for his guidance, feedback and engagement which have helped me better understand the passion, dedication and effort that go into research and science at the highest level as at LIGO. I am also grateful to him for giving me the opportunity to work on tasks not strictly related to the pith of my own research.

I am also grateful to Gautam Venugopalan for being a model of a graduate student I can aspire to be and whose constant presence and mentorship helped me better structure my work, set tangible goals and strive to achieve them. I would also like to express my sincere gratitude to Dr. Koji Arai for his infinite patience, unstinting help with all matters pertaining to my research and for teaching me the incredible value of careful documentation. I also thank him for sharing with me his wisdom and discussing his insights on life, science and research.

I also thank Prof. Alan Weinstein, Dr. Gabriele Vajente and Aaron Markowitz for their valuable insights and illuminating discussions. I am also grateful to my fellow SURF Kruthi Krishna for her efforts which were crucial to the progress of my own work and for her eager willingness to discuss at length the details of our projects and matters of scientific interest which made the late nights in the lab all the more enjoyable.

I would like to thank Prof. Karlheinz Brandenburg, Prof. Harishankar Ramachandran, Prof. Anil Prabhakar and Prof. Uday Khankoje all of whose support and wisdom made this experience possible. I am also grateful to the Student Faculty Programs office at Caltech, LIGO and NSF for this amazing opportunity. I am also extremely grateful to the facilitators of the S. N. Bose scholars program IUSSTF, WINStep Forward, Department of Science and Technology (DST), Govt. of India and the Science and Engineering Research Board of India for this opportunity. Finally, I would like to thank IIT Madras for encouraging research and enabling me to work at Caltech for the summer of 2019.

Contents

List of Figures

Abstract

LIGO interferometers used to detect gravitational waves achieve extremely high sensitivity through precise angular control of suspended optics that direct the laser beam. A host of sensing techniques, ranging from optical levers and wavefront sensors to suitably positioned quadrant photodiodes are used to detect the angular position and deviation of optics. This work attempts to introduce the use of Gigabit Ethernet (GigE) cameras capturing images of light scattered from optics to determine the position of the laser beam on the optic. A number of approaches based on tools from image processing are employed to discern the motion of the beam spot from video. They are found to be unreliable and discarded in favour of convolutional neural networks which can, in theory, learn any complex, non-linear mapping. These are trained on data generated at the 40m laboratory at Caltech and the results are analysed. Future work will rely on the use of data augmentation using conditional GANs to train networks and explore the utility of GANs for this task.

1 Introduction

Interferometers at LIGO possess extremely high sensitivity to be able to detect gravitational waves. The interferometers rely on being able to make extremely precise strain measurements using a laser beam resonant in the optical cavities of the interferometer. However, this also makes them highly susceptible to noise and makes keeping the system locked and in observing mode an incredibly challenging task. Keeping the system in lock and achieving high sensitivity requires that the laser beams in the system be incident at particular positions on the suspended optics which have been determined to be favorable. Even miniscule angular motion of an optic will lead to a shift in the position of the beam spot on the suspended optic. Deviation in the angular position from the desired set point can arise from various sources of noise ranging from tectonic and oceanic movements to nearby electronics to other human activities that couple into the suspension used to isolate the optics and cause movement along multiple directions.

Consequently, systems that can accurately detect the position of the laser beam spot on the optic need to be developed to be able to discern any displacement of the laser beam spot from the ideal position. Such measurements can in turn be used for precise angular control of the suspended optics such as the mirrors used in the interferometer. Several techniques such as the use of Quadrant Photodiodes (QPDs henceforth, see Figure 4) or angle to length coupling have been utilised to achieve this task of beam spot position determination. These are discussed in greater detail in Section 2. A QPD is placed behind an optic and laser power transmitted through the optic is incident on it. The position of the beam spot on the QPD can then be determined as described in Section 2 which serves as a proxy for the position of the beam spot on the optic. Similarly measurements of angle to length coupling can be made by applying a dither to the optic and studying the length signal [16] to determine the position of the laser beam spot. While highly effective, both these techniques have their drawbacks. For instance, the QPD can not be positioned behind all optics oftentimes due to their setup, one example being the beamsplitter. Further, to make angle to length measurements, one must dither the optic which injects noise into the system. Continuous measurement of beam position then requires constant injection of noise into the system which is not feasible.

In this work, an attempt is made to develop a non-invasive beam spot tracking system that relies on video data obtained using Gigabit Ethernet (GigE henceforth) cameras [10] set up to monitor a specific cavity. These cameras collect video data of light scattered from the surface of the optic due to surface roughness and point scatterers. Image processing algorithms are then developed to detect the position of the laser beam on the optic from the video data. These algorithms will take as input the video data collected by the GigE camera and output the position of the beam spot on the optic as a function of time. The GigE camera data provides information regarding the entire cavity in contrast to optical levers which, while highly useful, are local sensors and detect only the motion of a specific optic. Further these can be set up to monitor optics such as the beamsplitter the position of the laser beam on which can not be studied using a QPD. GigE cameras are used because they have very low latency and a high dynamic range[1].

This work focuses on using deep learning techniques to determine the position of the laser beam spot on the optic. Two simple techniques- centroid detection and contour detection- are used to determine the position of the beam spot. Their failure on real data collected from the set up GigE camera and the conclusions drawn prompt the transition to the use of Convolutional Neural Networks which in theory can learn any complex non-linear mapping. This document details the entire pipeline starting with data collection and concluding with the test of the neural network on unseen test data. Work done to simulate data to train neural networks on is also presented.

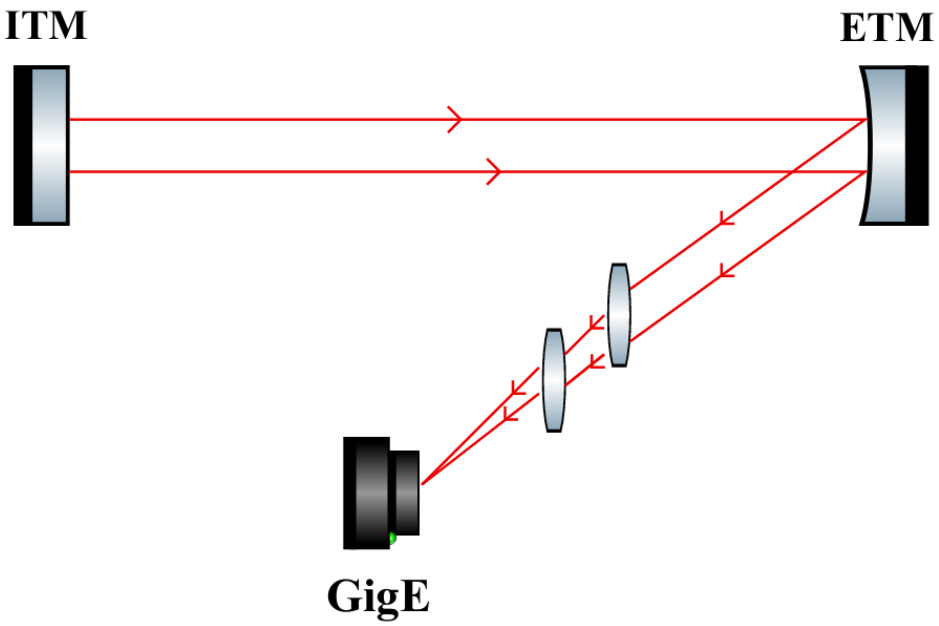


Figure 1: A schematic of a Fabry Perot cavity depicting the reflection of light off of a test mass and its subsequent focus and capture on the setup GigE camera

One extension of this work is to integrate the functional beam position tracking system into a feedback loop to control the angular position of the optics and center the beam spot. A typical control loop employing feedback control is illustrated in Figure 2. The dynamic

[1]https://www.baslerweb.com/en/products/cameras/area-scan-cameras/ace/aca640-120gm/

system or the plant is an electronic or mechanical component whose output needs to be monitored and made to match the input reference. In this particular case the plant is a suspended optical component such as a mirror whose pitch and yaw motion needs to be controlled. The feedback sensor is used to monitor the behaviour of plant and provide the information necessary to produce the desired control signal to the controller. In the case of the optical lever system used for angular control, the working of which is elucidated in greater detail in Section 2, this is a quadrant photodetector that tracks the position of the reflected laser beam. In this work, the GigE camera is proposed as an alternative sensor.

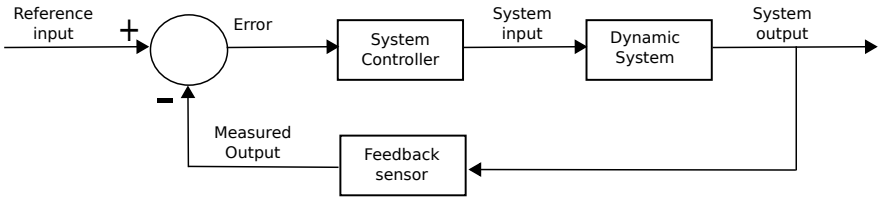


Figure 2: A typical feedback control loop

Section 2 presents the camera setup and more details of the functioning and use of QPDs. Following this, details of attempts to simulate data resembling real video data from the GigE are presented in Section 3. Section 4 presents the three approaches used for the purpose of laser beam spot tracking- two based on techniques using the Python library OpenCV [5] and the third using neural networks- and discusses the outcomes and attempts to provide explanations for the same. Extensions to this work are discussed in Section 5 followed by a few concluding remarks. It is worth noting that while the eventual goal is to discern motion of the order of a micrometer (ideally when no external force is deliberately applied to the optic and when it is ostensibly stationary), this work only accomplishes tracking of beam spot motion of the order of a millimeter and is consequently only a step in that direction. Therefore, Section 5 also presents untried techniques that can be used to advance this work.

2 Setup

The algorithms developed in this work take as input video data and process it frame by frame to output a series of position values corresponding to the X and Y coordinates of the beam spot on the optic. The first two approaches use remarkably simple techniques and are based on methods such as center of mass calculation, thresholding [13] and contour detection [14]. They require data in order for parameters to be tuned and to verify the effectiveness of the algorithm. The deep learning approach is inherently data driven and requires that a sizeable amount of data be collected to train CNNs effectively and to test their performance.

All the data used in this work is either simulated or obtained from the camera set up in the 40m prototype interferometer laboratory at Caltech. The setup of the camera is shown in Figure 3. The GigE camera used for this work is installed at a viewport of the MC2 optic of the Input Mode Cleaner (IMC) cavity. As seen in Figure 3, light scattered by point scatterers and due to the surface roughness of the curved optic is incident on an angled mirror which directs this into the viewport. The light then passes through a telescopic lens system that

focuses the image of the laser beam spot on the camera sensor. The exposure time of the camera can be adjusted using available software[2]. The data obtained from this GigE camera can be used as the input to any of the algorithms.

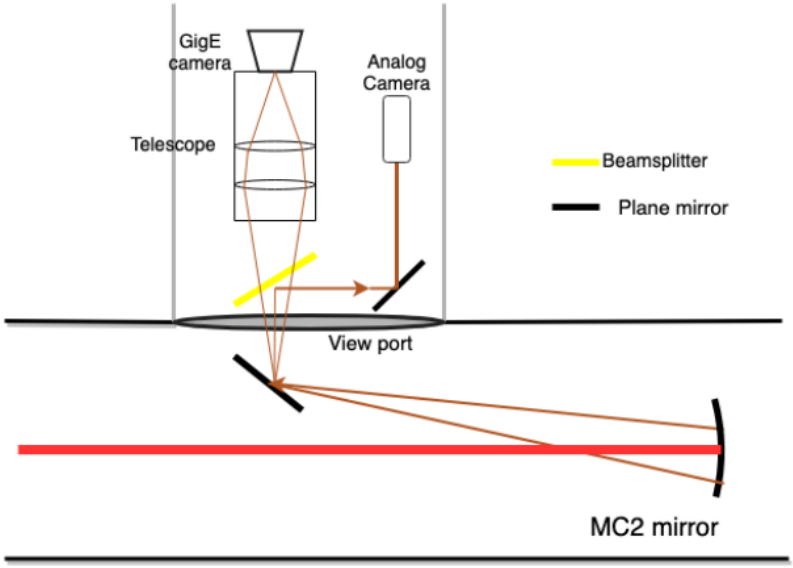


Figure 3: Setup of the GigE camera at the MC2 optic of the IMC

In the case of the deep learning based approach, it is also necessary to possess true labels which a network can be trained against. As described in Section 1, the readings of the QPD are used as the true position (or rather a proxy for it) of the laser beam spot. The quadrant photodiode consists of four photodiodes which are arranged to form the four quadrants of a coordinate system as shown in Figure 4.

If the incident laser beam produces outputs say A, B, C, D (see Figure 4) from the four quadrants, then they can be used to calculate the position of the beam using the following equations [8]

$$x = \frac{B + D - A - C}{A + B + C + D} \tag{1}$$

$$y = \frac{A + B - C - D}{A + B + C + D} \tag{2}$$

where x and y are the X and Y coordinates of the laser beam spot with respect to the QPD coordinate system. The time series of these x and y values corresponding to the frames of the video forms the labels of the training data. Section 5 discusses the need for an alternate source of true labels.

In the preliminary portion of the work, data collected in [10] is used to train CNNs. Here, optical lever (Oplev) readings are used to train the networks as opposed to QPD readings.

[2]https://git.ligo.org/40m/GigE.git

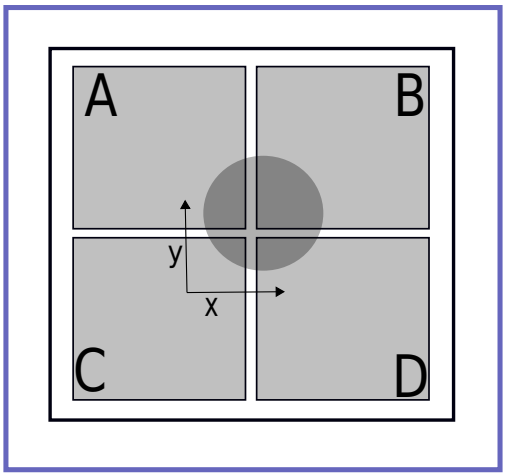


Figure 4: Illustration of a four quadrant photodiode

Optical levers capture the angular motion of a specific optic. Figure 5 depicts a simple optical lever consisting of a fiber coupled diode laser, quad photodetectors to determine beam location and displacement and structural pylons to mount the necessary hardware.

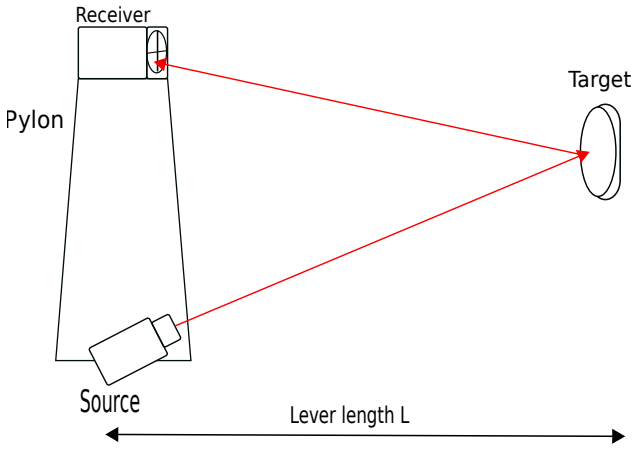


Figure 5: Design of an optical lever

In an optical lever, a laser beam is projected onto the suspended mirror and the system structured such that the reflected laser beam is incident on the QPD. The system is calibrated such that the output is zero when the beam is centered perfectly on the optic and the mirrors are aligned. However, if the suspended mirror rotates through an angle θ, then the reflected beam rotates through an angle 2θ. This is shown in Figure 6.

The position of the reflected beam can then be obtained using Equations 1 and 2 which help determine the position/deviation of the suspended optic.

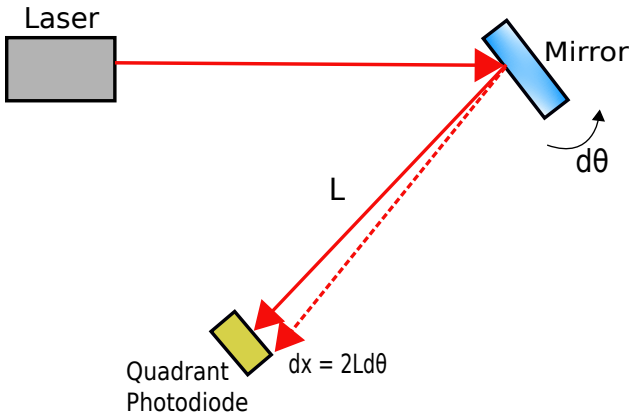


Figure 6: Angular deflection of the mirror and reflected beam [9] in an Oplev

3 Simulating a beam spot

As discussed in Section 1, scattering results in the production of a beam spot on the camera sensor which can be used to determine the angular motion of the suspended optic. The laser beam is expected to display Gaussian intensity variation. While abundant data is available for image processing, it might not always be labelled and it might sometimes just be more convenient to be able to generate data at will. This is why an attempt is made at simulating data. Videos of the beam spot motion are simulated by generating successive grayscale frames consisting of pixels whose intensity varies as a two dimensional gaussian with a constant variance for a given simulation and whose mean varies in a predetermined manner. Figure 9 shows two such simulations at different frame resolutions.

(a) (b)

Figure 7: Frames from the simulated data at resolutions of (a) 64 x 64 (b) 256 x 256

The motion of the beam spot is centered about the middle of the frame and can be sinusoidal or random. Figure 8 provides an example simulated motion of the beam spot using four constituent sinusoidal signals.

Further, uniform random noise is added to the images or frames to make the data similar to real data obtained from various cameras set up. While this constitutes image level noise, the motion of the beam spot itself can possess some inherent randomness. This is accounted for by adding numbers drawn from a uniform distribution to the mean of the gaussian in every

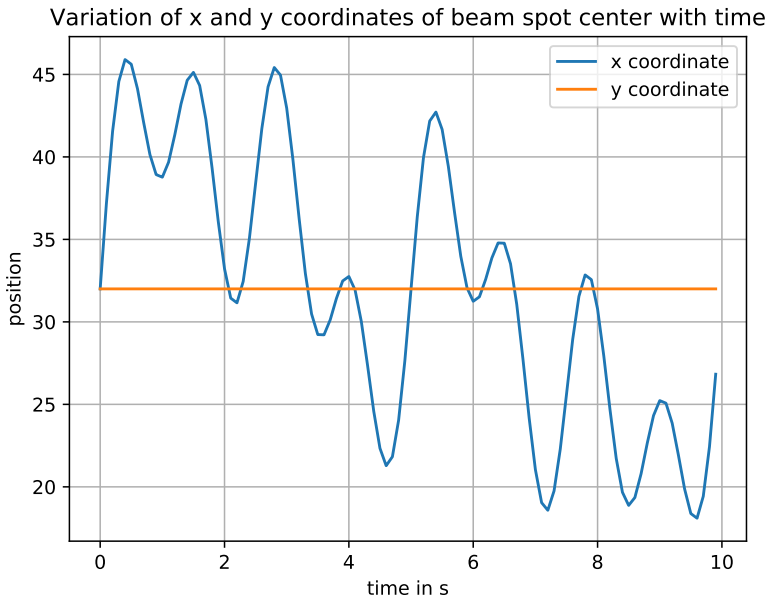
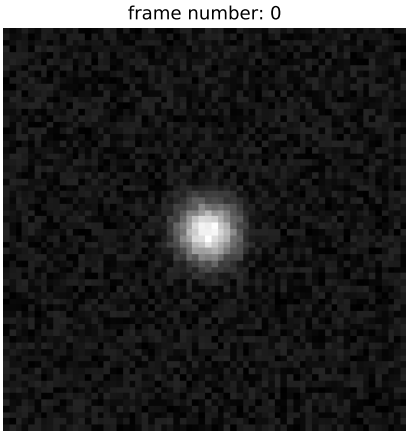


Figure 8: A plot of the motion of the center of the beam spot in which the y coordinate is not varied and the x coordinate follows motion caused by the combination of four sinusoids

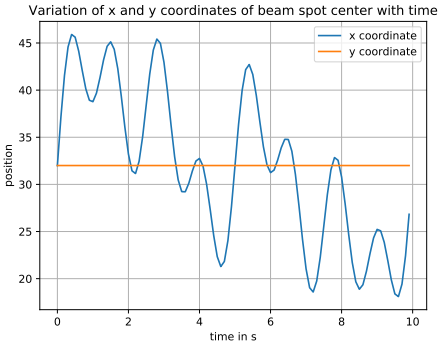
frame thereby introducing some randomness in the motion. Figure 9a results from adding noise (random numbers generated between 0 and 40, which is about a sixth of the maximum intensity value of 255) to the image in Figure 7a. Figure 9b depicts the motion after addition of image level noise and Figure 9c the motion of the spot center after introducing randomness into beam spot motion. All the simulation is done using the NumPy [6], Matplotlib [7] and OpenCV libraries.

Figure 10 includes frames from actual data of the beam spot that needs to be tracked. The video data from which Figure 10a was obtained is saturated and was only used to briefly test the method described in Section 4.1.2. The data from which Figure 10b was obtained was collected as part of the work done in [10]. Before the completion of the setup in Figure 3, this data was used to test the methods presented in Sections 4.1.2 and 4.2.4 and perform preliminary checks for the written code. However, all the results presented henceforth are based on the data whose frames look like Figure 10c. This data was obtained from the setup in Figure 3 as described in Section 4.2.2. All three types of available video data are presented here to draw a comparison between the nature of the data simulated and the actual data.

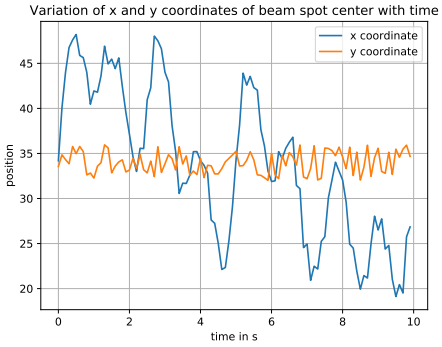
It is evident that the efforts presented here are simply a basic attempt to simulate the physical system. A more sophisticated simulation must capture a highly non-linear and complex process accounting not only for Gaussianity of the laser beam intensity profile but also the surface roughness of the optic which is characterized by the BRDF. Consequently, this must also take into account the angles of incidence of the laser beam and the angles of observation. It must also factor in point scatterers present on the surface of the optic which lead to bright spots in the captured image. The effect of point scatterers is characterized by the total integrated scatter which is modelled as a series of impulses. This discrepancy between the



(a)

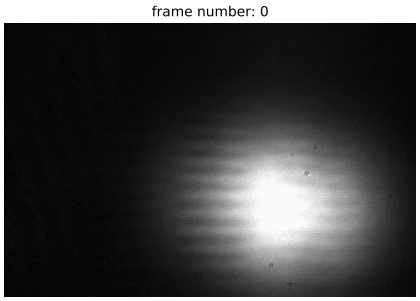


(b)

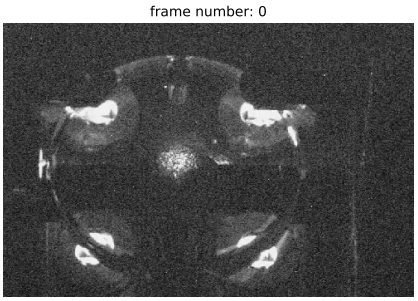


(c)

Figure 9: (a) A frame from the video created after the addition of noise (b) Variation of the coordinates of the center with only image level noise (c) Variation of the coordinates of the center with both image level noise and randomness in the motion



(a)



(b)

(c)

Figure 10: Frames from actual data

simulated and real data is particularly significant when evaluating the performance of the tracking algorithms as they perform extremely well on easier, simulated data but fail or need to be modified extensively to perform well on real data. An alternative to this approach of hard coding the physics of the problem to obtain simulated data is discussed in Section 5.

4 Beam spot detection

The problem of the spot detection can be tackled in multiple ways. The first is to use traditional image processing tools which are motivated and discussed below. Such methods are simple, interpretable and do not require massive amounts of labeled data. However, a degree of hard coding is required and even after performing well on simulated data, they may not transfer to real data in an obvious manner. The second approach is data driven and employs neural networks to detect the motion of the beam spot. The network is trained on individual frames of the video of the beam spot motion with the labels being the position of the beam spot in that frame. This approach is however very susceptible to overfitting. The following section describes the use of traditional algorithms to detect the beam spot motion.

For all the attempted approaches the signal to noise ratio (SNR) is used as a metric to evaluate the performance of the algorithm. The SNR is computed as the ratio of the power in the actual (true) position time series to the power in the error between the true and predicted values and is defined as below

$$SNR = \frac{\sum_i y_i^2}{\sum_i (y_i - \hat{y}_i)^2} \tag{3}$$

where y_i is the actual position at the i^{th} time instant and $\hat{y}_i$ the predicted position. In the case of the techniques described in Section 4.1.1 and 4.1.2, the predicted labels are with respect to the image frame coordinate system and are determined in terms of number of pixels. For the sake of comparison both the time series are normalized to one as can be seen in Figure 11 before computing the SNR. In the case of the CNN based method, the true labels (from the QPD) lie between 0 and 1 in arbitrary units. A separate experiment[3] was carried out to determine the calibration factor between the output of the QPD and the actual displacement of the laser beam spot on the optic. While the predictions themselves lie between 0 and 1, the calibration factor is used to determine the true displacement. The higher the SNR, the better the performance of the algorithm. The SNR tends to infinity when the true and predicted values match exactly.

4.1 Traditional image processing

Two methods are used to determine the centroid of the beam spot. Both these techniques are not data driven and do not make use of neural networks. They both compute the position of the centroid with respect to coordinate system of the image frames and it is assumed that

[3] https://nodus.ligo.caltech.edu:8081/40m/14804

knowledge of the geometry of the system will allow for calculations of the position of the laser beam spot on the optic.

4.1.1 Centroid detection

From observation of Figures 9a and 10 (grayscale images), it is evident that the intensity of the beam spot is centered around a particular spot which serves as an approximate centroid of the beam spot. Therefore, the first method attempted was a simple center of mass calculation. Here, the centroid is computed as a weighted sum of the pixel coordinates with the pixel intensity values serving as weights. Dark pixels will automatically not contribute to the calculation as they have values close to 0. The X and Y coordinates of the centroid can be computed as given below:

$$X_{centroid} = \frac{\sum_{pixels} X_{pixel} \times (\text{pixel value})}{\sum_{pixels} (\text{pixel value})} \tag{4}$$

$$Y_{centroid} = \frac{\sum_{pixels} Y_{pixel} \times (\text{pixel value})}{\sum_{pixels} (\text{pixel value})} \tag{5}$$

This was applied to data (of Figure 10c) at three different stages- raw unprocessed data, data on which median blur [12] has been applied and finally data processed using a median blur and then thresholding [13]. Applying median blur to an image with kernel size of n replaces each pixel with the median value of the pixels lying in an $n \times n$ square centered around the pixel. Thresholding as used in this approach replaces all pixels above a specified pixel value by 255 and those below it by 0. This algorithm is tested on data which is collected as described in Section 4.2.2. The results are presented in Figure 11.

The SNR improves slightly with the application of both thresholding and median blur. However, the predictions are a very poor approximation of the true labels as is visually evident. This is corroborated by the fact that the SNR is quite low- in comparison to the results obtained in Section 4.1.2 and 4.2.

4.1.2 Contour detection

The intensity variation of the beam spot is Gaussian and in the simulation, it is perfectly so. Therefore, it follows that the intensity contours of the spot are circular. Consequently, tracking the centroid of the beam spot reduces to determining this contour and then tracking the centroid of this contour. This can be done in the following sequence of steps. First, the images are converted to binary by thresholding. Here, a global value of threshold (127) is used as the threshold. All pixel values greater than this threshold are pulled up to 255 and the rest suppressed to 0. This results in a binary image useful for contour detection. This results in a circular contour that is used to determine the centroid of the beam. Initially, attempts were made to detect the circular boundary using Hough circle transform. However,

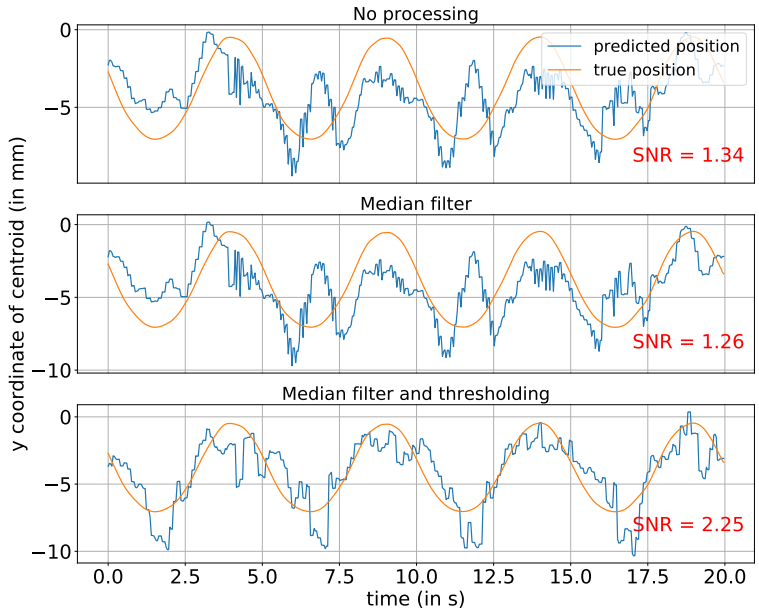


Figure 11: Center of mass tracking for GigE video from MC2 viewport with an applied dither of 0.2 Hz with three kinds of preprocessing

this approach resulted in either no detection or detection of spurious circles. Consequently, contour detection algorithms in OpenCV were used to determine the contour of the beam spot.

The centroid can then be found and this is the center of the gaussian beam spot. This works well in the case of simulated data, particularly in the case of data without noise as can be seen in Figure 13. As is evident, there is no error in prediction in the case of the y coordinate of the center. There is an error of $\approx 2\%$ in the predicted x coordinate. The predicted and actual coordinates were normalized for the sake of comparison. This also gives us an idea of the noise threshold in the case of simulations without noise. One can expect the algorithm to make perfect predictions when a video in which the spot is absolutely still is provided. This data corresponds to the simulation of Figure 7a

Adding noise introduces error in the predictions. Error is simply defined as the difference between the predicted and actual motion once both have been normalised to one. It does not affect the prediction of the x coordinate along which there is well defined motion but has significant impact on the prediction of motion along the y direction along which no motion is simulated. This can be seen in Figure 14 which corresponds to the simulation in Figure 9a. The error subplot for the x direction gives one the impression of there being large error. This is however misleading as it corresponds only to an error of at most 2 pixels in the actual prediction.

Finally, Figure 15 provides details of the tracking when the motion of the beam spot itself is slightly random. Here, numbers drawn from a uniform distribution between 0 and 4

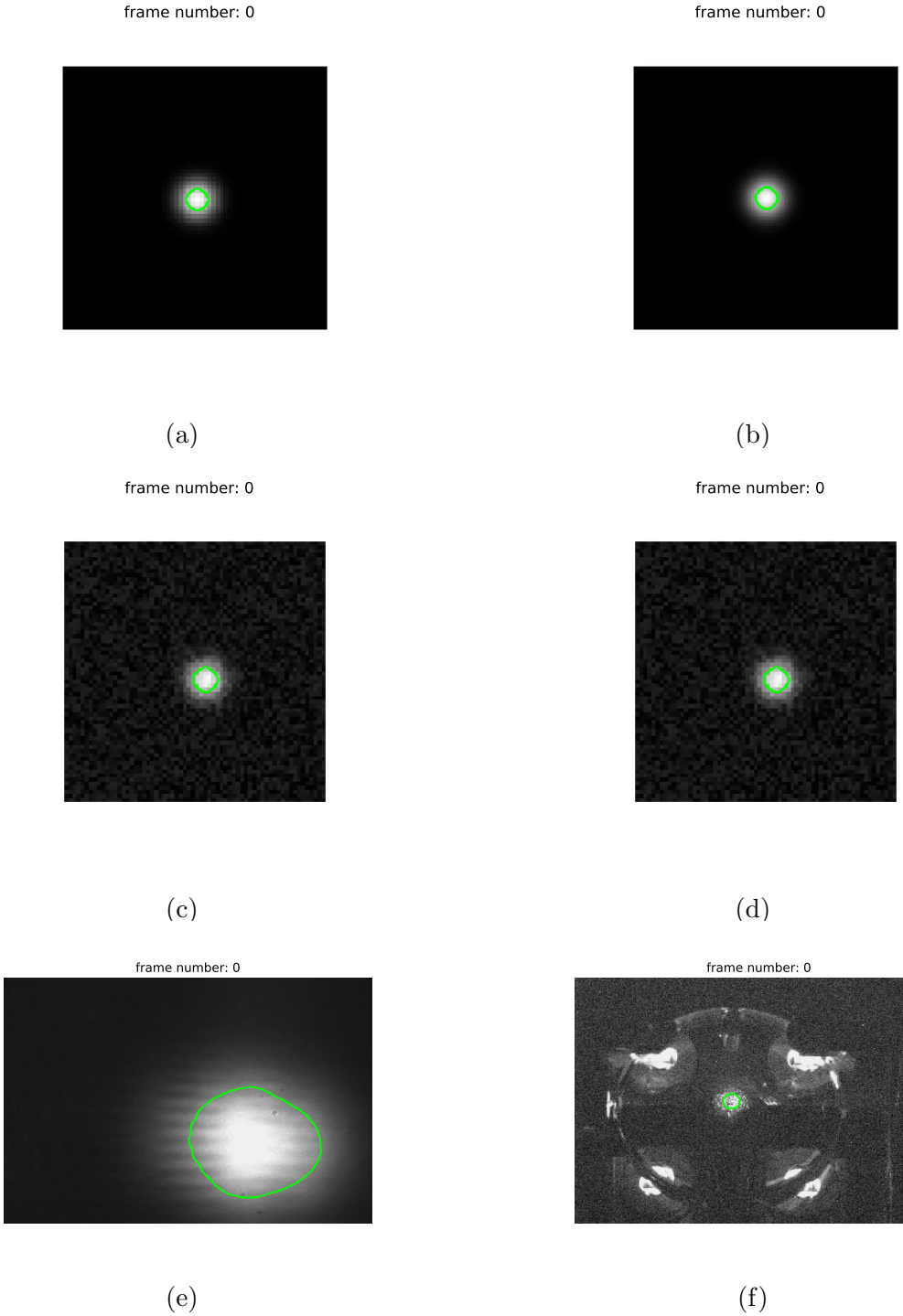


Figure 12: Contour detected in the video corresponding to frames in (a) Figure 7a (b) Figure 7b (c) Figure 9a (d) first frame for simulation with both image level noise and randomness in motion (e) Figure 10a (f) Figure 10b

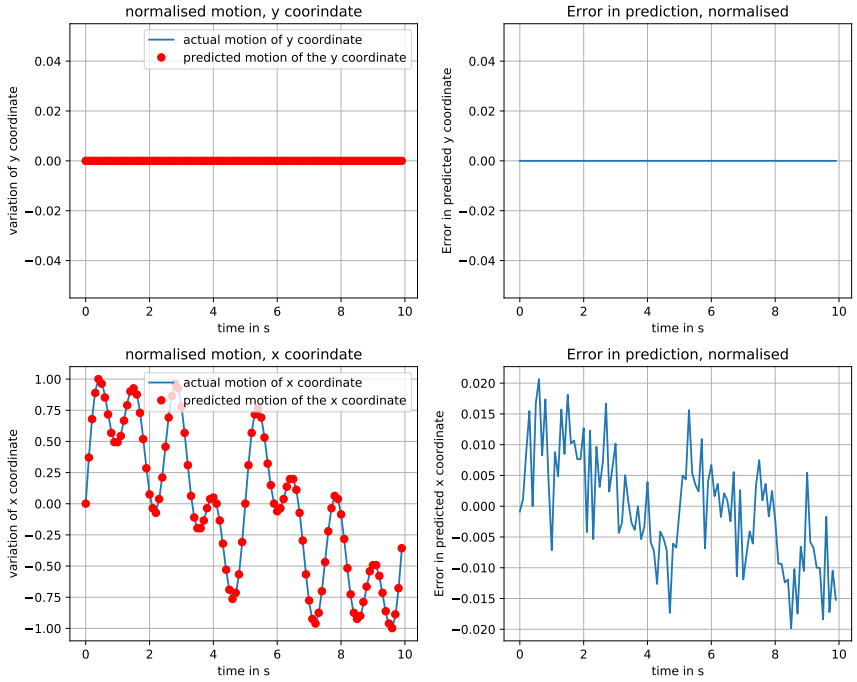


Figure 13: Comparison of the actual and predicted motion for simulated data

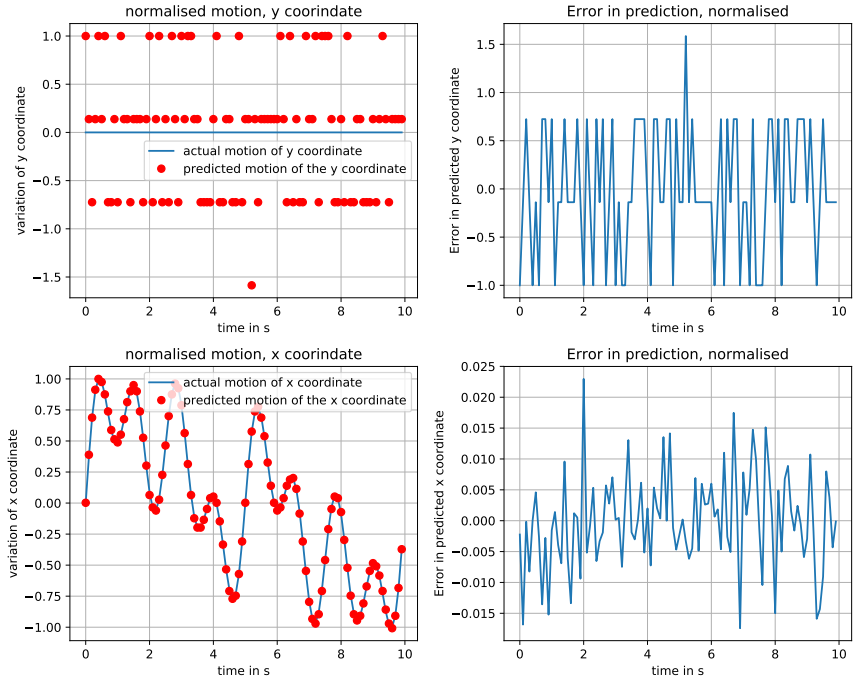


Figure 14: Comparison of the actual and predicted motion with image level noise for simulated data

(nearly an eleventh of the amplitude of motion, 45, as can be seen from Figure 8) are added to the mean of the gaussian beam spot at every time instance. As can be seen from the superimposed graphs of the motion, the algorithm does a fairly good job of tracking the position of the beam spot. It is worth noting that the tracking of the algorithm becomes significantly worse as the amplitude of motion decreases.

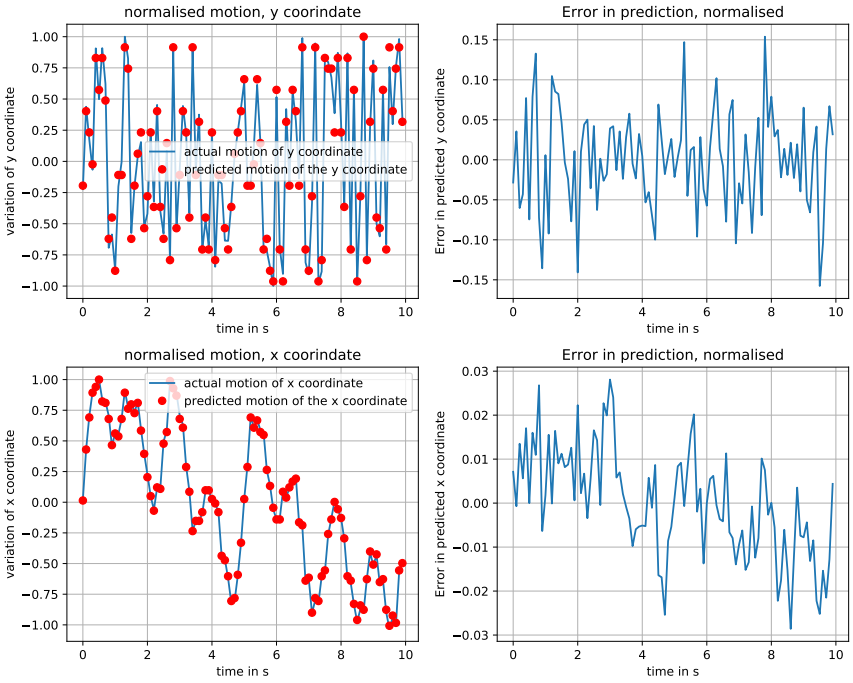


Figure 15: Comparison of the actual and predicted motion with image level noise and randomness in motion for simulated data

This technique was also tried on real data as shown in Figure 12f. The motion of the beam spot is expected to be sinusoidal along the Y direction corresponding to the pitch motion of the optic. However, the predicted motion resembles sinusoidal motion only superficially and further displays a phase difference with respect to the original motion. This is made use of in Section 4.2. Further, the predicted motion displays an unexpected degree of digitization in that there exist many groups of consecutive frames for which the algorithm determines the same position of the centroid. This is surprising and merits attention considering that the centroid is computed as the ratio of the first moment and zeroth moment and therefore is expected to vary smoothly.

Finally, this method does not work for the data of Figure 10c. In fact, no relevant contours were detected when this data was used. Other operations such as dilation and erosion had to be used to detect the necessary contours. It is this failure that motivates the next body of work.

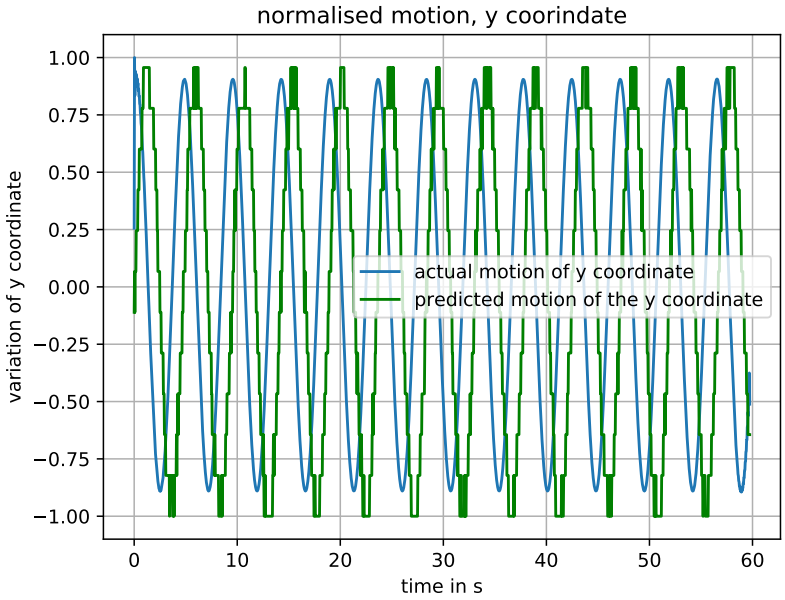


Figure 16: Performance of the proposed algorithm on noisy real data (presented in Figure 12f) corresponding to pitch motion of optic, position units normalized to make the comparison between oplev readings and algorithm predictions possible

4.2 Neural networks for beam tracking

Results like those presented in Figures 11 and 16 indicate that simple, unsophisticated techniques will not suffice for the task of determining the position of the beam on the optic from a GigE camera feed of scattered light. This mapping is a complex function dependent on several processes as described in Section 3. The aforementioned techniques fail for several reasons. The first being the assumption of Gaussianity. While it is true that the intensity profile of the laser beam is Gaussian, following reflection by the surface of the optic and due to point scatterers, the profile only remains approximately Gaussian which is no longer sufficient for the above methods to function robustly. Further, these techniques were only successful with saturated data and large amplitudes of motion. This is of little consequence if the motion to be detected is of the order of a few micrometers. In such cases, subpixel accuracy is required and can perhaps be achieved using the information presented by intensity variation of pixels rather than the motion of the cluster of bright pixels seen in Figure 10. Theoretically, such a complex and obscure mapping can be captured by a neural network. Figures 17 and 18 present the pipeline for training and testing a neural network.

4.2.1 Neural networks

Neural networks are, at the simplest level, mappings from a set of inputs x to a set of outputs y. Training a neural network involves learning a mapping f such that

$$y \approx f(x; W, b) \tag{6}$$

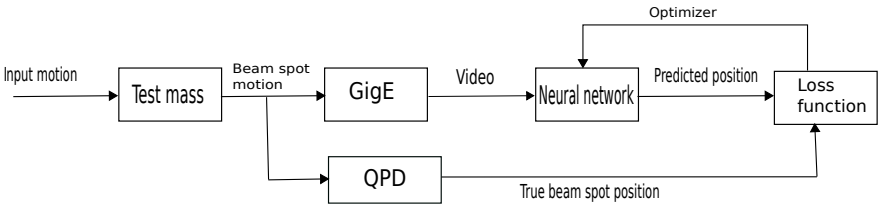


Figure 17: Training a neural network to determine angular motion

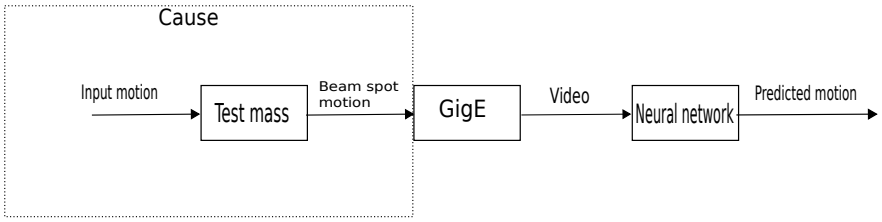


Figure 18: Neural network at test time

where W and b parametrize the function. As the mapping f is only approximate, an objective function is used to determine the goodness of this approximation. In the case of beam spot tracking which is a regression task, mean squared error defined as

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - f(x_i))^2 \tag{7}$$

where n is the number of examples/data points under consideration. The parameters which determine the mapping are obtained by treating this as an optimization task in which the objective function is minimized. In this work, this is done using the optimization algorithm ADAM [4]. Further, convolutional neural networks are used for the task of laser beam spot tracking as they are better suited for image processing tasks than are feedforward neural networks [15]. Some efforts to use feedforward neural networks for this task are presented in [10].

While neural networks can produce excellent results, this is contingent on the type and amount of data collected for the training. The details of the data collected are presented below.

4.2.2 Data collection

The data collected was primarily video data of the scattered light from the MC2 optic with a frame resolution of 640×480. Exposure time was set to 500μs. This was done to ensure that the image was not saturated and was verified by counting the number of pixels with an intensity value greater than 240 and ensuring that the percentage of the same did not exceed 1.5%. A plot of the saturation percentage variation with time is presented in Figure 19 for a fraction of the time for which the data was collected. Selection of the exposure time is an iterative process. Too high an exposure time leads to saturation and bleeding which might

then lead to loss of finer details such as variations in intensity which correspond to miniscule beam spot motion. Too small an exposure time leads to certain artefacts not being caught by the CCD sensor at all.

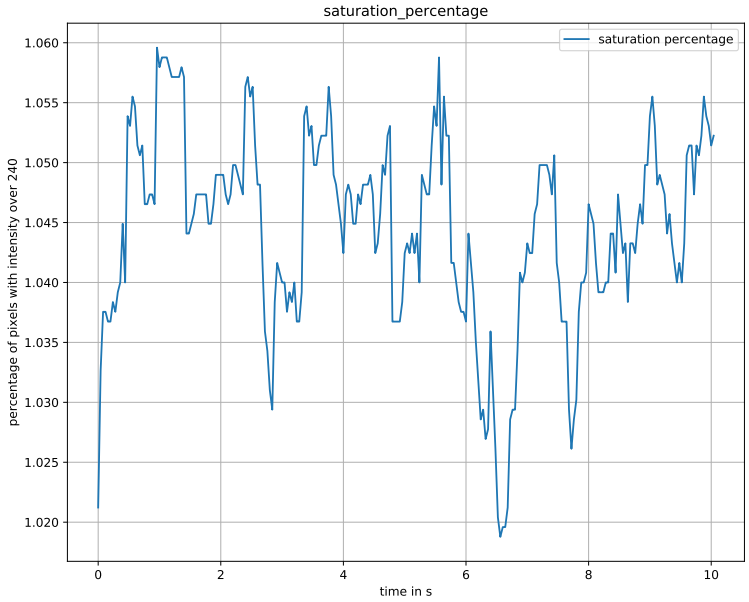


Figure 19: Percentage of pixels with intensity over 240 for 10s of the training data

In order to collect data, the MC2 optic is first given a sinusoidal dither. While an ideal tracking system should be able to detect motion at all frequencies and amplitudes however small, this work uses only a dither of 0.2 Hz. Data is collected for a total of 210s at 25 FPS. The X and Y coordinate readings from the QPD suspended behind the MC2 mirror shown in Figure 3 are used as the true labels for the data. The readings lie between 0 and 1. These are obtained from the channels "C1:IOO-MC_TRANS_PIT_ERR" and "C1:IOO-MC_TRANS_YAW_ERR". These are 1kHz channels and yield 1000 readings per second. However, in order for these to serve as the labels for the individual frames, the entire time series is resampled to 25Hz.

The data is fed frame by frame to the network and a prediction of the beam spot position is made. Consequently, the label at the i^{th} time instant must correspond to the frame recorded at the same moment. Therefore, it is necessary to synchronize the frames with the QPD readings. Two techniques were attempted for this. The first involved applying the contour detection algorithm described in Section 4.1.2 and then discarding values from the predicted and true value time series until the power in the error signal is minimized. If y_j is the j^{th} value in the series of true values of the beam position (y-coordinate) following the resampling and $\hat{y_k}$ the k^{th} predicted position, then the above procedure amounts to finding an m such that

$$m = \underset{m}{\operatorname{argmin}} \frac{1}{n-m} \sum_{i=0}^{n-m} (y_i - \hat{y}_{i+m})^2 \tag{8}$$

where n is the number of time instants for which data is available. A negative m implies the removal of samples from the start of the predicted series and end of the original series. Such a technique allows for two signals to be aligned very closely (compare Figures 16 and 20).

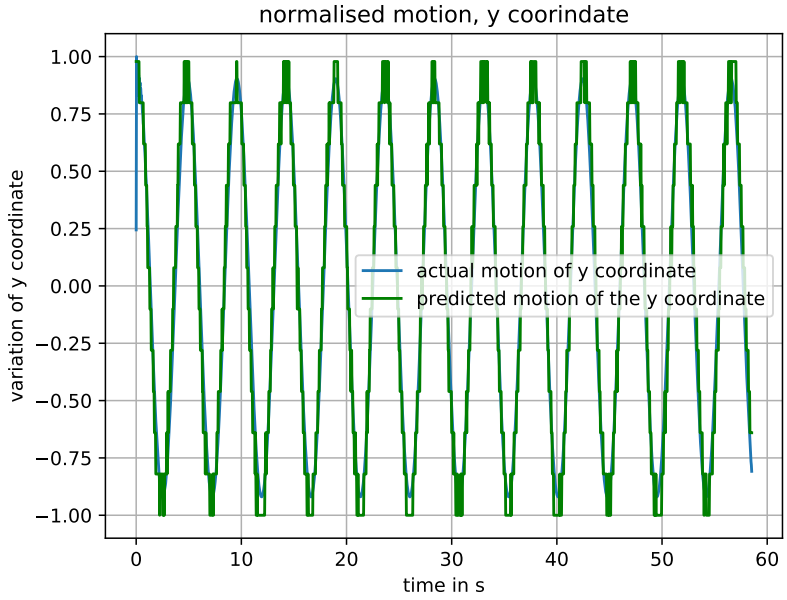


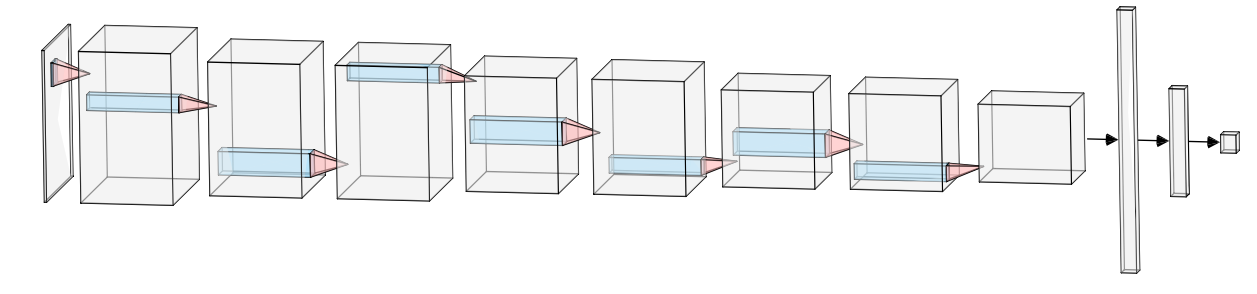
Figure 20: Predicted and actual motion after advancing the predicted signal by 36 samples, compare with Figure 16

However, it is very ad hoc and has some patent flaws. The first is that the underlying contour detection algorithm is not very robust. The parameters for thresholding and other operations need to be tuned every time new data is encountered. Consequently, this method is not used for the final synchronization. As an alternative, the GPS time was noted for the first frame of every video recorded and the aforementioned channel value readings obtained from that time instant. This is more versatile as it is possible to record the GPS time with very high precision irrespective of the kind of data being collected. Some more methods to tackle this synchronization issue are presented in Section 5. Both normalized (by 255) and unnormalized data were used for the network training and testing. Surprisingly, unnormalized data produced better results. Each image frame in the video was cropped to 350×350 before being fed to the network. This crop size was estimated by visually observing the cropped video frames and ascertaining that the beam spot lay, even at the peak of its motion, within the cropped frame. A more robust method of cropping is desirable, particularly if feeding in 640×480 frames without dimensionality reduction techniques such as PCA is not viable.

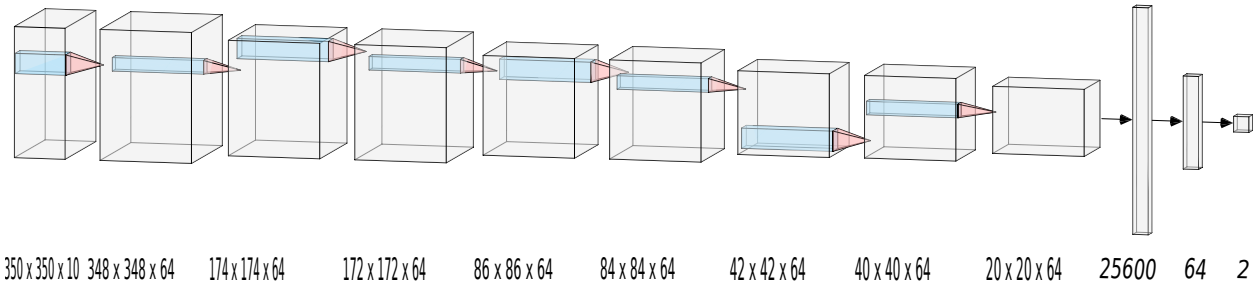
This work did not use simulated data for the training of neural networks. [10] presents several experiments involving training of neural networks on simulated data.

4.2.3 Architectures

The architectures experimented with are the ones presented in [11]. These primarily consist of alternating convolutional and max pooling layers. The nonlinearity used was the rectified linear unit (ReLU). The output and penultimate layers are feedforward layers, with linear and ReLU activations respectively. A schematic for the architecture is presented in Figure 21.



(a)



(b)

Figure 21: CNN architectures for beam spot tracking with (a) framewise input (b) input volumes

Both CNN architectures have an output size of 2- the X and Y coordinates of the beam spot on the optic. The CNN architecture presented in Figure 21a takes as input a single frame of the video to be processed and then predicts the corresponding position values. However, it is also worth noting that the movement of the beam spot can only contain certain frequency components. The physics of the system ensures that, under normal circumstances, erratic jumps do not occur in the position of the beam spot. Therefore, as shown in the architecture of Figure 21b, the CNN is supplied with multiple frames as input with the hope that the network will learn to capture the fact that high frequency components can not be present in the beam spot motion. The CNN architecture presented in Figure 21b takes as input 10 consecutive frames and predicts as output the position corresponding to the foremost frame.

While using CNNs that take as input a series of frames is one way of introducing memory into the system, another approach is to use LSTM networks which treat the beam spot tracking as a sequence to sequence task with the input being, once again, a sequence of video frames and the output being either a timer series or a single value corresponding to one of the frames of the video. The architecture used was an encoder decoder model with a CNN encoder that converts each image frame into a vector embedding that is then fed to a LSTM unit. Experimentally, these were much harder to train and produced results

(as measured by maximum deviation from true value and SNR) inferior to the simple CNN based approach.

All networks are implemented in Keras with a TensorFlow backend. Dropout before each of the dense layers and l2 regularization for the convolutional layers are used to avoid overfitting. Batch normalization is not used while training the networks.

4.2.4 Training the networks

The networks are trained on a GPU (initially a GTX1060 then TitanX) to tune hyperparameters. The hyperparameters that can be tuned in this case are the learning rate, the batch size, dropout ratio, number of epochs of training, type and size of preprocessing filter used (usually median blur), number of frames in the input volume. A grid search is performed to determine the optimal set of hyperparameters for the learning task. The results are presented in Section 4.2.5.

4.2.5 Results

The network with the best performance from Table 1 is tested on 20s of test data. In Table 1, hyperparameters such as dropout ratio, convolutional filter size appear to be constant for all trials. However, these too were subject to the grid search and best results were obtained at the presented values. The predictions for the Y coordinate position are presented in Figure 23. The SNR obtained for this is much higher than that obtained from the center of mass computation. However, it is only 1dB greater than the results in Figure 22 that obtained using the contour detection method. However, this method is much more general and holds the promise of high accuracy even for small amplitudes of motion. While the motion tracked here is of the order of a millimeter, more data and better hyperparameter tuning can possibly lead to tracking of the motion of the order of a micrometer.

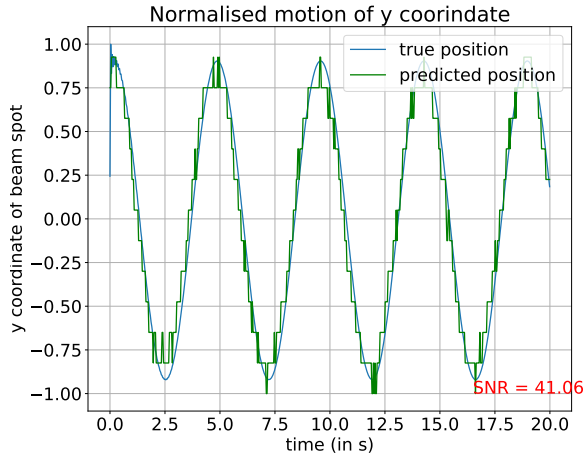


Figure 22: Predicted motion of the beam spot along the Y axis using the method described in Section 4.1.2, same as Figure 20

Trial	Batch size	Dropout probability	Learning rate	Filter size (median)	Memory size	Epochs	Kernel shape	Test error power in dB
1	32	0.5	0.0001	1	10	75	(3, 3)	-23.22
2	64	0.5	0.0001	1	10	75	(3, 3)	-21.03
3	64	0.5	0.0001	1	5	75	(3, 3)	-20.47
4	32	0.5	0.0001	1	1	75	(3, 3)	-16.82
5	32	0.5	0.0004	1	5	75	(3, 3)	-20.51
6	32	0.5	0.0001	1	5	75	(3, 3)	-21.35
7	64	0.5	0.0004	1	10	75	(3, 3)	-21.75
8	32	0.5	0.0004	1	1	75	(3, 3)	-14.35
9	64	0.5	0.0001	1	1	75	(3, 3)	-17.96
10	32	0.5	0.0001	1	10	75	(3, 3)	-23.22
11	64	0.5	0.0001	1	5	75	(3, 3)	-20.47
12	32	0.5	0.0007	1	10	75	(3, 3)	-22.70
13	32	0.5	0.0001	1	10	75	(3, 3)	-22.99
14	32	0.5	0.0001	1	5	50	(3, 3)	-19.34
15	32	0.5	0.001	1	10	50	(3, 3)	-21.45
16	64	0.5	0.0001	1	5	50	(3, 3)	-15.64
17	64	0.5	0.0001	1	10	50	(3, 3)	-17.26
18	32	0.5	0.001	1	1	50	(3, 3)	-14.27
19	64	0.5	0.001	1	5	50	(3, 3)	-18.42
20	32	0.5	0.0001	1	10	50	(3, 3)	-20.27
21	32	0.5	0.0001	1	1	50	(3, 3)	-16.90

Table 1: Results of performing a grid search for hyperparameter tuning for architectures of Figure 21 to determine the network which produces lowest test error

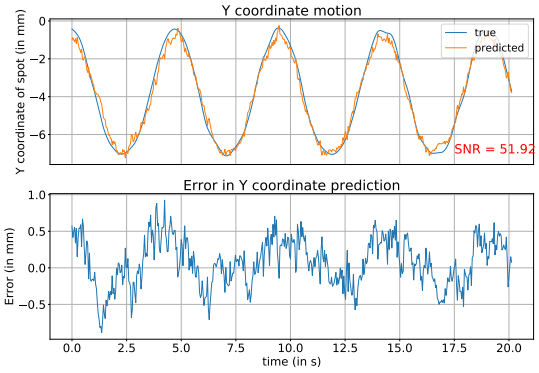


Figure 23: Predicted motion of the beam spot along the Y axis

The learning curves for the training of the same network are presented in Figure 24. The decline in the training loss indicates that the network continues to learn with each epoch. However, for the corresponding choice of hyper parameters, the validation loss begins to increase at the 25^{th} epoch or so indicating that the network is beginning to overfit. This can be addressed using a host of techniques discussed in Section 5.

5 Future work

From Figure 24, it is evident that there is some degree of overfitting. While dropout was used as a part of this work, this can perhaps be rectified by using techniques such as early stopping or L2 regularization for the weights. While these options are available, a far more suitable course of action is to simply collect more data. The presented network was trained on 3.5 minutes of training data collected within a span of an hour from the GigE camera set up at the MC2 optic viewport as described in Section 2. However, this data corresponds to motion at 0.2 Hz and at quite a large amplitude of about 3 mm. When the input to the network is single frame as in Figure 21a, the frequency of motion ought not to matter as the CNN is learning to predict a position from the pixel values of an image at a given time instant. When fed an input volume as in Figure 21b and expected to capture frequency information of the motion, training the network on data collected over a range of frequencies is likely to increase its applicability.

A simple method to test this hypothesis is to first test the trained network discussed in Section 4.2.5 on data collected at the same exposure time but with different dither frequencies and amplitudes. It is expected that the network from Figure 21a will show no significant difference in performance irrespective of the frequency shift as it has no memory whereas the network which takes in an input volume will not generalize as well. Both networks are expected to fail when it comes to smaller amplitudes of motion. On observing the results of this experiment, new and much more data at different frequencies and amplitudes needs to be collected for training. Ideally, the network ought to be trained on data collected when no external force is applied to the optic.

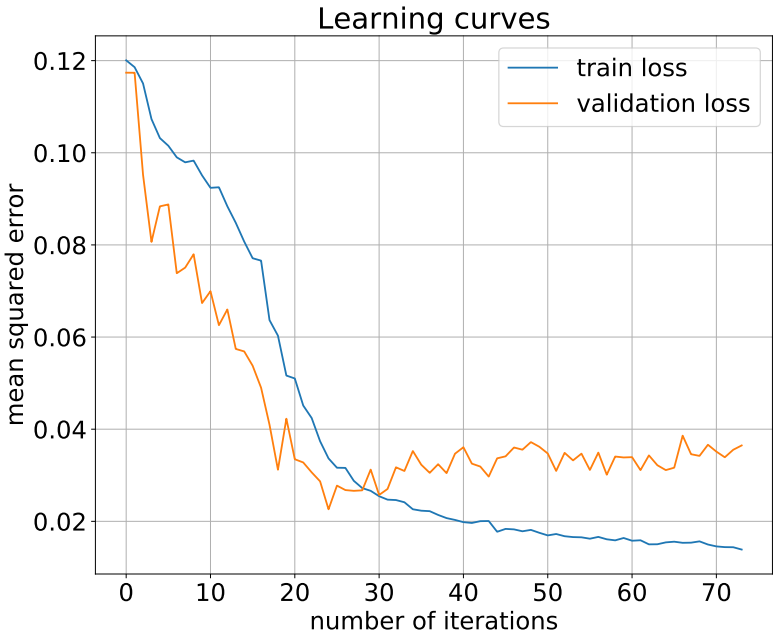


Figure 24: Learning curves for the training of the network whose output is shown in Figure 23

Another interesting question worth considering when collecting data is the suitability of the QPD readings as the true labels. A network trained on these readings will inevitably be only so good as these. One alternative is to use A2L readings to determine the position of the beam spot on the optic as described in [16] and train a network against the same. This is bound to improve the quality of the predictions of the network.

Further, the analysis presented in this work assumes that the data collected for network training at one point in time is representative and accommodates all possible situations. This is not necessarily true. For instance, this is based on the assumption that the set up GigE camera does not move over time. When desiring high precision, even slight motions of the GigE camera will change the nature of video data produced by the GigE camera and therefore the output of the network. This can be tackled in two ways. The first is to alter the nature of data used to train the network. From Figure 10c it is clear that the image is zoomed in onto the beam spot. Alternately, data with the camera zoomed out can be collected to capture features such as the OSEMs (optical shadow electromagnets) in the image as has been done in Figure 10b. This might provide the network with points of reference with respect to which the position of the beam spot can be determined. An alternative is to collect data while moving the camera itself by small amounts to simulate data generated when the mount of the camera is subject to motion.

Another method to create more data, specially to accommodate those situations for which data has not been or can not be collected in a short amount of time is by enhancing the simulation. The processes that need to be considered to enhance the simulation are described in Section 3 and must factor in surface roughness, point scatterers and angles of observation.

The endeavor is to identify all the parameters that will create images as close to that obtained from the GigE as possible. This data can then be used to train the network.

Another method to create data is to use generative adversarial networks (GANs) which have produced remarkable results in generation tasks and have successfully been used for data augmentation. GANs consist of a generative network which generates data and an adversarial network which distinguishes the generated data from the real data. These networks are trained competitively and the end result is a network that takes in a noise vector to generate image data. The problem of generating GANs for beam spot tracking requires the use of conditional GANs. These take in a deterministic vector and generate data, in the simplest sense, corresponding to that. Here, we want the generator to take in a 2 dimensional vector of X and Y coordinates and then output an image frame resembling that which would be produced by a GigE camera of a beam spot in that position. While they are tricky to train, this effort will help identify what the limitations of GANs for this task are.

One issue discussed in Section 4.2.2 was that of ensuring a correct mapping between the frames of the video and the data collected from the QPD. Two methods were presented for this- one rather ad hoc method involving the use of the method described in Section 4.1.2 and another in which the GPS time for the first frame was recorded and the latter was used throughout the work in Section 4.2. However, a far more reliable method is to synchronize the clock of the GigE camera and that of the interferometer. This will allow the camera to capture a frame at every clock edge maintaining synchronicity.

Further, the hyperparameters for the network presented in Section 4.2 need to be tuned to assess if such architectures are suitable at all for this task before being dismissed. Other architectures worth trying are those in which the input is a three dimensional volume on which 3D rather than 2D convolution is performed. This too introduces some memory into the system and might just help the network capture the idea that the beam spot can not move at very high frequencies. One application of a beam position tracking system is in control of the angular position of the optics of the interferometer a schematic for which is presented in Figure 25. This requires that the network be implemented on suitable hardware to achieve the necessary speed following which it needs to be integrated into existing or new feedback loops to achieve precise control.

6 Conclusion

This work explored the problem of laser beam position tracking using a camera feed from Gigabit Ethernet cameras. The use of cameras in addition to existing methods was motivated and the setup used for this work detailed. Multiple approaches for beam tracking were attempted and it was determined that the use of neural networks is most appropriate for this task. Neural networks are trained with hyperparameters tuned using a grid search and beam spot motion at 0.2 Hz with an amplitude of about 3mm is tracked with maximum error under 20%. Following this, the results are analysed and possible courses of action to expand upon this work identified.

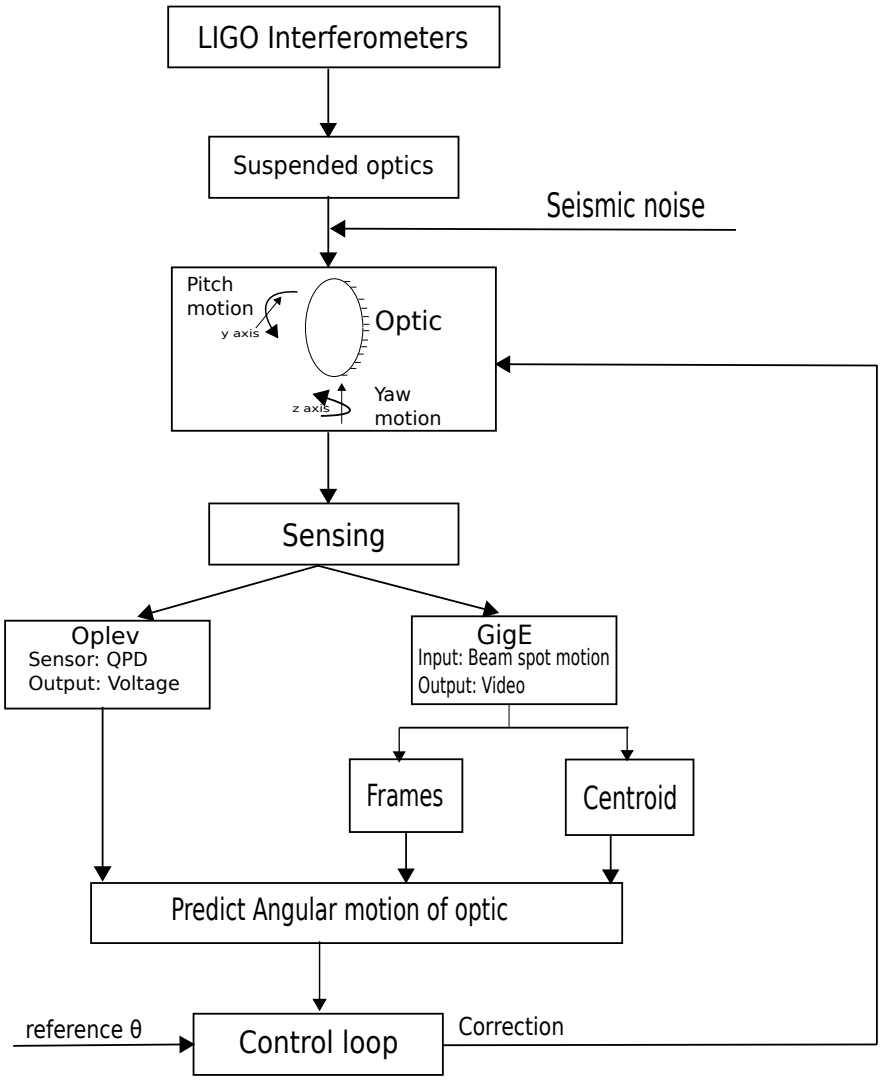


Figure 25: Schematic depicting the scheme for angular control of suspended optics using local sensors (optical lever system) and global sensors (GigE cameras)

References

[1] Rana Adhikari, *Sensitivity and noise analysis of 4 km laser interferometric gravitational wave antennae*. PhD diss., Massachusetts Institute of Technology, 2004.

[2] Ryan DeRosa, Jennifer C Driggers, Dani Atkinson, Haixing Miao, Valery Frolov, Michael Landry, JosephGiaime, Rana Adhikari. *Global Feed-Forward Vibration Isolation in a km scale Interferometer*. Class. Quantum Grav. 29 215008, 2012.

[3] Eric D. Black, Tara Chelermsongsak, Riccardo Desalvo,Zach Korth, Mark Barton, Doug Cook, Cheryl Vorvick,Hiro Yamammoto, Giovanni Salvi, Rob Schofield, Michael Smith. *Advanced-LIGO Optical LeversDesign Requirements*. Internal working note, LIGO. 2010.

[4] Diederik P. Kingma, Jimmy Lei Ba. *Adam: a Method for Stochastic Optimization*. arXiv preprint arXiv:1412.6980 (2014)

[5] Bradski G. *The OpenCV Library*. Dr Dobb's Journal of Software Tools. 2000.

[6] Travis E, Oliphant. *A guide to NumPy*. USA: Trelgol Publishing, (2006).

[7] John D. Hunter. *Matplotlib: A 2D Graphics Environment*. Computing in Science & Engineering, 9, 90-95 (2007).

[8] https://www.aptechnologies.co.uk/support/SiPDS/bicell-quad

[9] https://dcc.ligo.org/public/0115/G1401146/002/Lecture_Control.pdf

[10] https://dcc.ligo.org/public/0152/T1800245/002/final_report.pdf

[11] https://dcc.ligo.org/DocDB/0153/G1801377/002/Neural_nets.pdf

[12] https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_filtering/py_filtering.html#median-filtering

[13] https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_thresholding/py_thresholding.html

[14] https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_contours/py_contours_begin/py_contours_begin.html#contours-getting-started

[15] http://cs231n.github.io/convolutional-networks/

[16] https://dcc.ligo.org/public/0128/T1600397/002/BeamPositionFluctuations.pdf