Utilizing Motion Sensor Data For Some Image Processing Applications

A Project Report

submitted by

SARAGADAM R V VISHWANATH

EE10B035

in partial fulfilment of the requirements

for the award of the degree of

BACHELOR OF TECHNOLOGY

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY MADRAS

MAY 2014

THESIS CERTIFICATE

This is to certify that the thesis titled **Utilizing Motion Sensor Data For Some Image Processing Applications**, submitted by **Saragadam Raja Venkata Vishwanath**, to the Indian Institute of Technology, Madras, for the award of the degree of **Bachelor of Technology**, is a bona fide record of the research work done by him under our supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. A. N. Rajagopalan
Research Guide
Professor
Dept. of Electrical Engineering
IIT-Madras, 600 036

Place: Chennai

Date: 12th May 2014

ACKNOWLEDGEMENTS

I would like to express my heartfelt gratitude to my guide, Dr. A. N. Rajagopalan for his invaluable support and guidance throughout my project duration. I have gained immense insight into computational photography and image processing in this one year, which I believe will hold me in good stead in my future endeavors. I am indebted to him for patiently sitting with me at times when I could not see any path forward. I will always remember him as the professor who said “There is nothing sacred about time or frequency”.

I also thank all my lab mates Vijay, Abhijith, Sahana, Swarun, Karthik and Poorna for being extremely helpful and for their patience and guiding me through difficult tasks. I had the privilege of working with some of the best minds in the department.

I would also like to thank my parents for being flexible and giving me moral support throughout my education. I believe that what I am today and what I will be tomorrow is a result of their wonderful upbringing. I am also thankful to my brother, who constantly updates me with the latest trends in technology and comes forward to give constructive critisism on my programs. His hobby of photography has also proved very useful in understanding some of the nitty-gritties of my project.

Last but not the least, my wing mates have been the best experience in my undergraduate life. Living with fifteen different minds with always something new to learn from each person, I have learnt more than I ever learnt anything before coming here. They have been with me throughout my undergraduate studies, with healthy debates, joyful times, tough exams and quintessential assignments. I owe a lot to them.

Vishwanath

ABSTRACT

We explore various image restoration and registration techniques using data obtained from a mobile device. The first one is non-blind image deblurring. We evaluate the usefulness of having extra information from the motion sensors, which give a clue about camera motion. We show that deconvolution with our kernel as the initial estimate either converged quicker or produced better results for low texture images when compared to a complete blind deconvolution. The second problem we address is that of depth estimation using blurred and latent image pair and depth from focus. In the former problem, we show that with a good estimate of blur kernel from motion sensors, depth can be estimated very efficiently. The third problem involves evaluation of image registration algorithms using the information from motion sensors. We show that the registration results are very accurate and the registration can be done very fast and efficiently even when one of the images is blurred.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

There has been a recent shift in computation from traditional computers to ubiquitous mobile phones. With the ever increasing power of the mobiles, executing many of the image processing algorithms on mobile has been made easy. Further, with the many peripherals available for the mobile, such as the inertial sensors, controllable focus, TCP and the bluetooth, computational photography is greatly advanced.

Photography using mobile has the innate problem of shake, which tends to blur the images. Recovering the sharp image, given only the blurred image is a difficult task due to the ill-posed nature of the problem. Any information about the motion during the exposure time can help solve the problem with ease. Data from motion sensors serves this purpose by providing the approximate trajectory of motion. Even when taking multiple photographs which need to be aligned, like in the case of HDR images from multiple normal exposure images, sensors data can be utilized to find the accurate motion between the two shots.

A recent advancement in the field of computational photography is light field photography. Instead of a single focused image, multiple images are taken and various regions of the image can be refocused. A mobile platform, which has the ability to capture multiple images very fast with variable focus would pave way for research in this area.

We evaluate some of the important algorithms in computational photography by using data obtained from the mobile. The aim is to prove that the extra information available from the motion sensors aides in increasing the efficiency of the image processing algorithms.

1.1 Prior work

The best known project involving the camera as an imaging device is the Camera 2.0 project by the Computer Science department at Stanford. The project involved hacking the Nokia N900 firmware to write custom algorithms for focus. Further, the lab also built a custom camera called Frankencamera which runs Linux and is completely programmable, making it highly flexible.

While computational imaging on a mobile platform is relatively new, there has been a lot of work on combining motion sensors with DSLR to perform accurate deblurring. Joshi *et al.* (2010) Combines a 3 axis accelerometer and 3 axis gyroscope with a DSLR. This paper is one of the main motivation for the image deblurring part. Bae *et al.* (2013) builds upon Joshi *et al.* (2010) and explains how blind PSF estimation can be combined with motion sensors data to improve deblurring. Šindelář and Šroubek (2013) discusses the usage of a smart phone for deblurring images which have been degraded by rotational blur. We try to use some of the ideas in this paper for deblurring for translational blur.

Depth estimation, a well researched topic is of a lot of interest, especially in devices for gaming like Kinect. A lot of literature exists on depth estimation using stereo vision. A recent foray into this area is to estimate depth from a sharp and blurred image pair. While most of the methods estimate it by estimating PSF from individual regions, not much work has been done on fusing the image data with motion sensors data. We present some of the methods for estimating depth using motion sensors data along with sharp and blurred images.

Image registration exists for sharp images but not much literature exists on registering blurred images. Registration for symmetric blur has been discussed by Ojansivu and Heikkila (2007), which is more applicable to the case of defocus blur. We deal with a more non-symmetric case of motion blur and show that inertial sensors gives accurate registration results.

1.2 Report layout

The project report is organized in the following manner. The chapter on *Basic Theory* presents a brief account of the various algorithms and the mathematics involved in the project. The chapter on *The Device* gives an in-depth account of the mobile platform, the application written for the mobile platform and the host side software. The chapter on *Deblurring* explains the implementation of the deblurring algorithm using the data obtained from the mobile. *Image Registration* discusses about how inertial sensor data can be used to reduce the search space of image registration process. *Depth Estimation* has two parts, *Depth estimation using motion blur* and *Depth estimation using focus measure*.

At the end of the report, we give a brief discussion about the future of the project and how the algorithms can be taken forward. We concluded that though some of the experiments were not successful, they gave very interesting results, which pave way for future work.

CHAPTER 2

BASIC THEORY

We discuss the basics of some of the image processing algorithms in this chapter to gain an understanding of the theory behind the project.

2.1 Position from Acceleration Data

A major part of the project involves estimating the trajectory of motion from accelerometer data. In the current context, we assume that we have access to a three-axis accelerometer alone. The three-axis accelerometer measures physical acceleration, which is a combination of the acceleration due to gravity and any external force. Further, the measured values are relative to the mobile frame of reference. Define a_x, a_y, a_z as the acceleration due to external force, g_x, g_y, g_z as the acceleration due to gravity from the mobile's frame of reference and $\hat{a}_x, \hat{a}_y, \hat{a}_z$ as the measured acceleration. Hence, measured acceleration is modeled as

$$\begin{aligned}
\hat{a}_x &= g_x + a_x + n_x \\
\hat{a}_y &= g_y + a_y + n_y \\
\hat{a}_z &= g_z + a_z + n_z,
\end{aligned}$$

where n_x, n_y and n_z are noise in measurements.For calculating the trajectory, we need $\{a_x, a_y, a_z\}$. To obtain these values, we need to subtract the acceleration due to gravity. However, we have no further information apart from the above equations. To get a good estimate of the required acceleration, we assume that the gravity vector is quasi-static when there is only translation in the mobile device. Hence, we estimate the gravity vector as

$$g_x^t = \alpha g_x^{t-1} + (1-\alpha)\hat{a}_x^t$$
$$g_y^t = \alpha g_y^{t-1} + (1-\alpha)\hat{a}_y^t$$
$$g_z^t = \alpha g_z^{t-1} + (1-\alpha)\hat{a}_z^t,$$

where α is the LPF constant and lies between 0 and 1. Higher α gives rise to slower variation in the gravity vector. We found empirically that 0.8 was a good value for the constant.

Once we have the device acceleration, the equation for finding the trajectory of motion would be

$$x(t) = \iint_0^t a_x(t)dt$$
$$y(t) = \iint_0^t a_y(t)dt$$
$$z(t) = \iint_0^t a_z(t)dt.$$

The discrete time approximation is

$$x[n] = (\sum_t^n \sum_t^n a_x[n]) \times T^2$$
$$y[n] = (\sum_t^n \sum_t^n a_y[n]) \times T^2$$
$$z[n] = (\sum_t^n \sum_t^n a_z[n]) \times T^2,$$

where T is the sampling time of the motion sensors. Though this gives a good estimate of the orientation of the device when there is no motion or the device acceleration when there is only translation, the method suffers from two drawbacks:

1. The noise in measurement creates a random walk pattern. However, this might not be of major concern as it would only create a small disturbance in the over all trajectory.
2. If the estimate of gravity vector is not correct, it would create a constant offset to the device acceleration. This gives rise to drift, which will grow as t^2, giving rise to wrong trajectory.

Fig.2.1 explains how the drift can be of serious concern when estimating the trajectory.

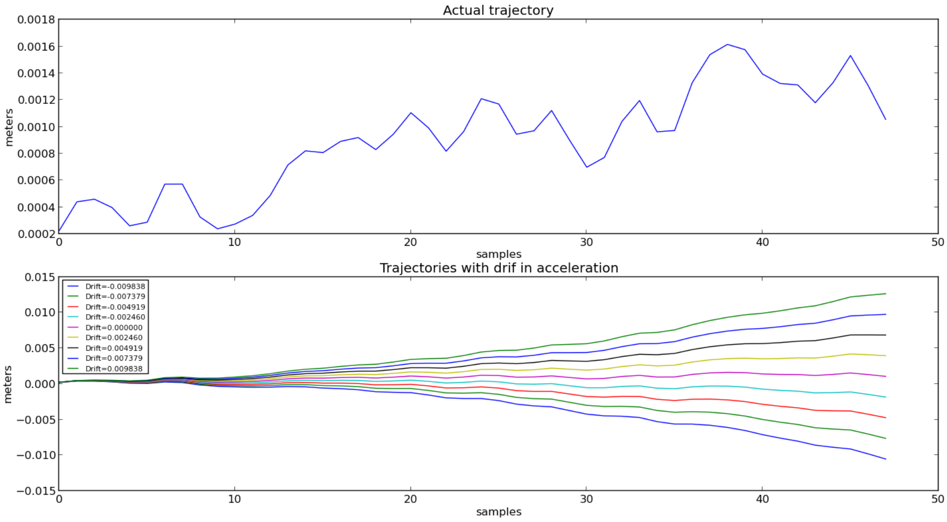

Figure 2.1: Plot showing the effect of constant offset to acceleration value. The drift was iterated from -1% to 1% of the maximum acceleration value. With even a small error in the estimate, the calculated trajectory goes off the actual trajectory.

Due to lack of information regarding the orientation of the mobile, which gives rise to a bad estimate of gravity vector, the drift will exist and make the estimate unreliable. In section 4, we will see the consequences of have a bad trajectory and possible ways to rectify the problem.

2.2 Image blur

Image blur is equivalent to low pass filtering in 1D signals. While image blur is desired in some applications like animation, it is undesirable when taking still photos. Two types of blurs are of interest in this project, namely defocus blur and motion blur.

2.2.1 Defocus blur

Defocus blur is caused when the image of the object is not focussed on the sensor. This arises only in real aperture cases when the focal length of the lens is finite and objects are at varied distances.

The blur that arises due to defocus can be modeled as a 2D gaussian filtering operation. Hence, the point spread function is given by

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

where σ depends on the depth of the object from the lens.

The defocus blur can be used in the shape from focus method, which varies one particular parameter of the system like the lens position or the object position to bring the object in focus. The idea is to have a *focus measure*. Pertuz *et al.* (2013) gives a good outline of various focus measures used commonly. Further discussion about shape from focus will be carried out in section 6.

2.2.2 Motion blur

Motion blur occurs due to relative motion between the scene and the camera during the exposure time. The motion could either be the moving object or the motion induced due to camera shake. This report deals with the problem of camera shake.

The process of camera shake can be considered as an averaging of multiple sharp images shifted due to the camera motion. Hence for a planar scene,

$$\hat{im}(x,y) = \frac{1}{N} \sum_{k}^{N} im(x - x_k, y - y_k)$$
$$\implies \hat{im}(x,y) = \frac{1}{N} \sum_{k}^{N} \delta(x_k, y_k) * im(x,y)$$

where $\delta(x_k, y_k)$ is the Kronecker delta and $\hat{im}$ is the observed image.

This can be rewritten as,

$$\hat{im}(x, y) = (\frac{1}{N}\sum_{k}^{N}\delta(x_k, y_k)) * im(x, y)$$
$$\implies \hat{im}(x, y) = h(x, y) * im(x, y)$$

$h(x, y)$ is called the point spread function. If the image is not a planar scene, then every point in the scene moves by a different amount due to the camera shake. If we assume that there is no parallax error, we have,

$$\hat{im}(x, y) = \sum_{t}\delta(\frac{x_t}{d(x, y)}, \frac{y_t}{d(x, y)}) * im(x, y)$$

where $d(x, y)$ is the depth of the image at (x, y). This space variant blurring operation can be used to calculate depth information from the images, as objects at different depths will be blurred to a different extent. Calculating $d(x, y)$ gives the depth map of the image.

2.3 Image deconvolution

The process of removing the blur in the image is known as image deconvolution. Deconvolution can be broadly of two types, blind and non- blind. If the PSF is known *a priori*, it is known as non-blind deconvolution, else, it is known as blind deconvolution. Blind deconvolution is an ill-posed problem, as an infinite combinations of latent image and PSF can give rise to the observed image. Hence, various constraints are used when trying to retrieve the latent imageFergus *et al.* (2006); Krishnan and Fergus (2009); Levin *et al.* (2007*a*); Gupta *et al.* (2010).For example, an image can be considered as piecewise smooth function with sharp images. Hence, we can try to reduce the total variance of the obtained latent imageMoney and Kang (2006); Chan and Wong (1998). If the blur is caused due to a straight line shaped PSF, analyzing the frequency domain of the image can help retrieving the PSFOliveira *et al.* (2007).

On the other hand, non-blind deconvolution is less ill posed than the blind case, as we have the PSF estimate. As discussed in the previous section, the blurred image can be represented as the convolution of the PSF and the latent image in case of a planar scene. This transforms into multiplication in the frequency domain. Hence, we divide the image's Fourier transform with the PSF's Fourier transform to get the original image. i.e

$$IM_{latent}(u,v) = \frac{I\hat{M}(u,v)}{H(u,v)},$$

where H(u,v) is the DFT of the PSF $h(x,y)$. However, the observed image is further degraded by noise. Hence, the actual model of the observed image is

$$\hat{im}(x,y) = h(x,y) * im(x,y) + n(x,y),$$

where $n(x,y)$ is the noise. The equation in the frequency domain can be loosely expressed as

$$\begin{aligned} I\hat{M}(u,v) &= H(u,v).IM(u,v) + N(u,v) \\ IM_{latent}(u,v) &= \frac{I\hat{M}(u,v)}{H(u,v)} \\ \implies IM_{latent}(u,v) &= IM(u,v) + \frac{N(u,v)}{H(u,v)} \end{aligned}$$

Note that $h(x,y)$ has a low frequency structure and noise has a high frequency structure. Hence, the obtained image will have very high values in the high frequency region due to the zeros of $h(x,y)$ in the high frequency region.

Wiener deconvolution

Instead of simple division, we reformulate our problem as finding a filter which when convolved with the observed image would return the latent image. Further, we wish to reduce the error between the latent image and the image obtained by filtering the observed image, i.e

$$\min \quad \mathbb{E}|IM(u,v) - F.I\hat{M}(u,v)|^2$$
$$I\hat{M}(u,v) = IM(u,v).H(u,v) + N(u,v),$$

where $\mathbb{E}|.|$ is the expectation operator. Differentiating with respect to F, we get

$$\begin{aligned} F(u,v) &= \frac{H^*(u,v)}{|H(u,v)|^2 + \alpha} \\ \implies IM_{latent}(u,v) &= I\hat{M}(u,v).F(u,v) \\ &= \frac{H^*(u,v).I\hat{M}(u,v)}{|H(u,v)|^2 + \alpha} \end{aligned}$$

where $H^*(u,v)$ is the complex conjugate of $H(u,v)$ and α is the inverse signal to noise ratio of the image.

2.3.1 Regularized deconvolution

Since we know that the image blows up at the edges, we can impose a penalty on the edge weights. Let g_x and g_y be the horizontal and vertical differentiation filters . Then, we need

$$\begin{aligned} minimize \quad |y - h * x|^2 + \alpha|g_x * x|^2 + \alpha|g_x * x|^2 \\ \implies minimize \quad |Y - HX|^2 + \alpha|G_xX|^2 + \alpha|G_xX|^2 \end{aligned}$$

Where X represents the DFT of x. Differentiating the above equation with respect to X, we get

$$X = \frac{H^*Y}{|H|^2 + \alpha(G_x^2 + G_y^2)}$$

This is called regularized deconvolution. Apart from the simple deblurring algorithms, there are other deconvolution methods, which work by reducing a certain cost function. Literature on some of these can be found at Long and Brown; Wang *et al.*

(2009); Bioucas-Dias *et al.* (2006)

2.4 Image registration

Image registration finds application in many areas like medical imaging, aerial photography and automated target recognition. In computational photography, image registration is used in wide number of areas like panorama stitching, depth estimation using stereo vision and video stabilization.

Image registration involves finding the transform between a pair of images. The transform could be a simple translation and rotation or a complicated warping. For simple cases which involves only translation or only rotation, many of the frequency based methods give good results. For advanced warping, feature based registration is used. The SIFT based image registration, for example, identifies key points in each image. Once we have the corresponding points in both the images, the problem is of estimating the matrix that created this transform.

Note that when there is a large motion between the two images or one of the image pairs is highly blurred, registration becomes difficult. Section 5 discusses about how registration can be simplified using data from accelerometer. In this section, we will look at a simple case of registering image pair which are transformed by shift alone.

2.4.1 Image registration using cross correlation

Let im_1 and im_2 be the image pairs. Since we are looking at a translation only model, we have,

$$im_2(x, y) = im_1(x - x_0, y - y_0)$$

Given these two images, we wish to estimate (x_0, y_0). A shift in the spacial domain reflects as a phase multiplication in the frequency domain. Hence, if IM_1 is the DFT of

im_1, then,

$$IM_2(u,v) = IM_1(u,v)e^{-j(\frac{2\pi x_0 u}{M}+\frac{2\pi y_0 u}{N})}$$

Let,

$$H(u,v) = \frac{IM_1(u,v)\times IM_2^*(u,v)}{|IM_1(u,v)\times IM2(u,v)|}$$
$$\implies H(u,v) = e^{-j(\frac{2\pi x_0 u}{M}+\frac{2\pi y_0 u}{N})}$$

This gives,

$$h(x,y) = \delta(x_0, y_0)$$

Hence, the location of peak in h(x,y) will give the shift between the two images.

CHAPTER 3

THE DEVICE

Primarily, we want the device to function as an acquisition system which would take pictures and measure some sensor values. While choosing the mobile platform, we considered the following features

- The device should be low cost. We are looking at a mobile which would not cost above Rs. 15,000. This gave us the option to choose Android mobiles or the low end Windows Phone Mobiles.
- The device should have a good Software Development Kit. Android and Windows Phone both have a very good SDK.
- The device should have accelerometer, gyroscope and manual focus ability.
- The device should have easy image handling capabilities. Here, Windows Phone was a clear winner due to the powerful Nokia Imaging SDK.
- The device should have TCP/Bluetooth capability. Again, every smart phone available today has these features.

With the above points in mind and considering the low cost, we chose to go for Nokia Lumia 520 which features a 5 MP camera, three-axis accelerometer, WiFi and Bluetooth support. However, the trade off was that we could not get a gyroscope on the mobile. This would be a serious hindrance for handling rotation of the mobile along with translation. However, since we were only evaluating the possibilities of computational photography on the mobile platform, we chose to evaluate the mobile platform with the accelerometer alone.

From the ease of usage perspective, the Windows Phone 8 SDK has a very simple interface with coding in C# and GUI design using XAML script. Visual Studio makes it very easy to create applications (apps) fast. A number of code examples for developing WP8 apps are available at Network (2013–)

3.1 Device side application

The app is mainly used for logging information to the computer so that algorithms can be tested off line. The major advantage of this method is that it gives the power of the computer along with the flexibility to experiment with different kinds of algorithms.

To make the application versatile and useful for future development, we added multiple features to capture images. A screenshot of the application is given in Fig.3.1.

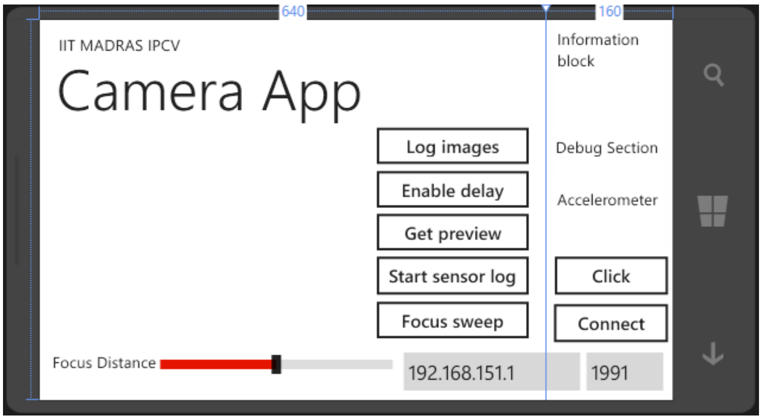

Figure 3.1: A screen shot of the application.

The top right, **Information block** serves as a debugging block during the image capturing process. For example, when taking images with long exposure, it indicates the number of samples of accelerometer data obtained. The **Debug section** box indicates the focus position and other debug information. The focus position can be controlled by the slide bar which is situated at the bottom of the application.

The **Accelerometer** box indicates the current value of the accelerometer. The button titled **Click** is used for capturing a single long exposure image. **Connect** button is used for connecting to the computer using TCP. The Host and the port can be selected in the fields at the bottom right of the application screen. Apart from these features, the following features add versatility to the application

- **Log images** enables continuous logging of the image frames to the computer when TCP stream is open. With the default setting, the duration between two frames is close to 20ms.
- **Enable delay** adds a 500 ms delay between the image retrieved from the image buffer and the long exposure shot.

- **Enable preview** captures an image from the image buffer (which can act as a latent image) prior to a long shot image.
- **Start sensor log** enables continous logging of the accelerometer data to the computer when the TCP stream is open.
- **Focus sweep** records 100 images with varied focus distance and sends them to the computer.

The application has a 10ms timer, which takes care of all the logging and image capture functions. While it is more apt to use a separate thread, we found that using a timer was an easy method.

We discuss some of the methods implemented for logging the data and sending it over TCP to understand the intricacies involved.

3.1.1 Logging accelerometer data during exposure

When the **Click** button is clicked, a camera capture sequence is started, which is programmed to take a picture with an exposure time of 200ms (adjustable). However, by default, this suspends all other thread activities and is only released once the capture is complete. This poses the problem that the accelerometer data cannot be logged during this time, which is very important for estimating the blur kernel . To overcome this, we use the async mode of camera capture(add ref to async camera capture website). In this mode, the camera is initialized for capture and it returns to the thread. Meanwhile, a busy flag is set and accelerometer data is recorded. When this is done, the busy flag is unset, which stops the data logging. The code snippet is given below

```
public async void capture(bool get_preview, bool register)
{
   // Get image preview
   if (get_preview == true)
   {
      _camera.GetPreviewBufferArgb(preview_image);
   }
   // Take a picture. Flag busy meanwhile.
   cam_busy = true;
   // If register mode is enabled, sleep for 500ms.
   if (register == true)
   {
```

```
        await System.Threading.Tasks.Task.Delay(500);
    }
    await _camsequence.StartCaptureAsync();
    cam_busy = false; // Done capture. Unset busy.
    transmit = true;
    imstream.Seek(0, SeekOrigin.Begin);
}
```

If an exposure time of 200ms is used, 20 accelerometer values are expected with a 10ms timer. However, close to 70 readings are obtained. This is due to the transfer time of the image data also included in the capture time. To overcome this problem, a fast moving image of a single point light source is taken. Frames of 20 samples out of 70 samples of the accelerometer data is used for estimating the best start and end times. We use these start and end times for all the future kernel estimates.

3.1.2 Sending data over TCP

The TCP communication enables a communication stream between the mobile device and the computer. With the stream open, it can be treated as a continuous information channel, with no prior information about the size of the data being sent. To retrieve data with this *asynchronous* type of data transmission, we encapsulate the data in keywords which is known to the computer. This would boil down the problem to searching for starting and ending keywords for each data type received. For example, if we are sending a new image, we would use the keywords **STIM** and **EDIM** to encapsulate the image data. An example:

```
// app_comsocket is the object representing the TCP
    connection.
// app_comsocket.Send is used to send data over the
    connection.
app_comsocket.Send("STIM\n"); // Start token
app_comsocket.Send(imdata); // Image data
app_comsocket.Send("EDIM\n"); // End token
```

3.2 Host side application

Most of the host side application is written in Python. The reason for choosing python is the vast support available for scientific computing, communication protocols, image processing support and above all, python being free software. However, some of the deblurring codes and super resolution are in MATLAB.

The communication from mobile to computer is done using the inbuilt *socket* module in python. As mentioned in section 3.1.2, the data from the mobile is encapsulated and sent in keywords. These keywords are used to parse the incoming data and split it accordingly.

A number of functions have been written as part of host side software for data reception, interpretation and image processing. Some of the main software modules are discussed below.

3.2.1 TCP data handling

The TCP data handling, as the name suggests is responsible for receiving data from the mobile, which includes images and motion sensors data. The main functions of this module involve receiving sharp and blurred image pair, continuously receiving images and continuously receiving motion sensor data. For example, in order to receive a sharp and blurred image pair, we need to execute the function `get_tcp_data()`, which in turn executes the following:

```
# Listen to the TCP socket and get data.
dstring = tcp_listen();
# Parse and save the data obtained from TCP
save_data(dstring);
# Return a handle to the data, which consists of images and
    the motion sensor data.
return TCPDataHandle(dstring)
```

For continuous reception of images, we use the following function calls.

```
# All the images are encapsulated in tokens. Further, all
    tokens are
```

```
# Separated by the null character.
strt_token = 'S\x00T\x00I\x00G\x00'
end_token = 'E\x00D\x00I\x00G\x00'
frame_token = 'S\x00T\x00I\x00M\x00'
fstart = 'STIM'
fend = 'EDIM'
# First, receive the data from the mobile.
continuous_recv(strt_token, end_token, frame_token,
   'tmp/burst/tokens.dat')
# Once we have all the data, split it and save the first 100
   images.
extract_images('tmp/burst/tokens.dat', 100, fstart, fend,
   'tmp/burst/src')
```

3.2.2 Visualizing sensor data

Another important feature of the host software is the ability to visualize the sensor data dynamically. This is very useful for understanding the attitude of the device and how the motion sensor data can be used for operations like video stabilization.

There are two main functions for visualizing the data. The first function shows a continuous graph of the measured accelerometer value. A screenshot of the graph is shown in Fig.3.2.

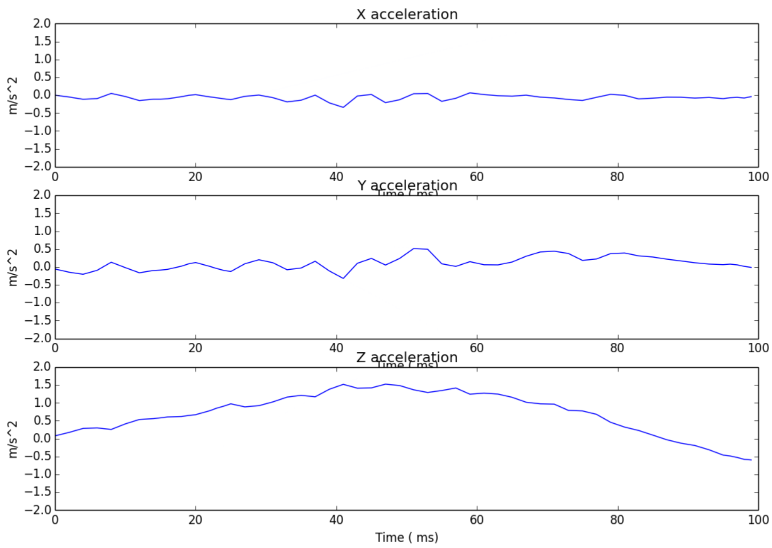

Figure 3.2: Screen shot of live sensor logging.

The second function is used for visualizing the attitude of the mobile for in plane motion. With the availability of only the accelerometer, we created a UI for showing the orientation and the jerk direction, obtained from the velocity vector, both corresponding to in plane motion. Fig.3.3 an example screen shot.

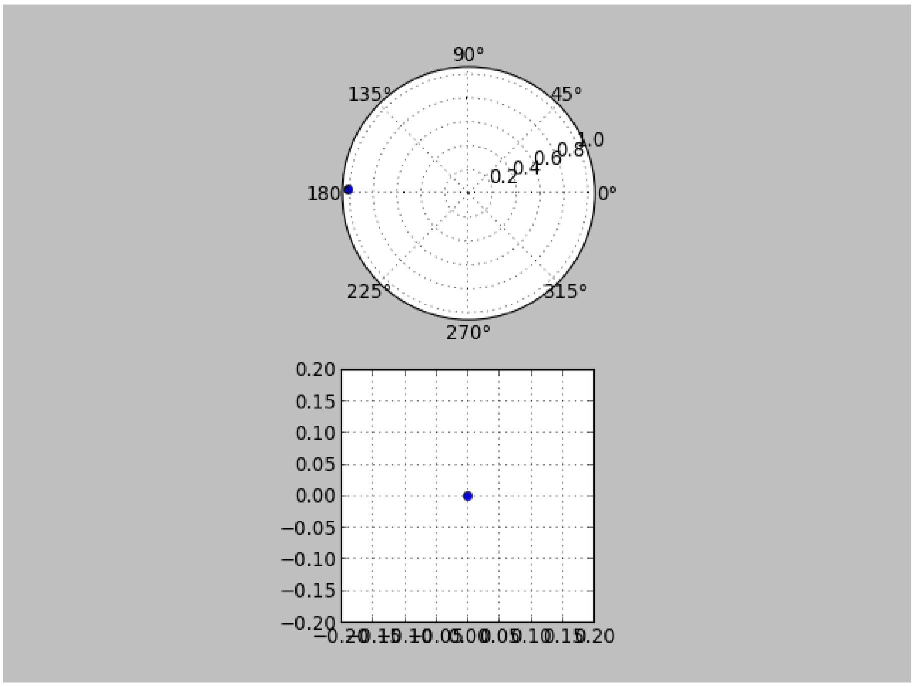

Figure 3.3: Screen shot of attitude UI. The upper circular graph represents the orientation of the mobile device and the lower rectangular graph represents the direction of translational motion.

CHAPTER 4

DEBLURRING

As mentioned in section 2.3, blurring due to motion is not desirable and we wish to remove it. In this section, we will look at non blind and semi blind deconvolution techniques.

Prior to deblurring, we need to estimate the blur kernel. As mentioned in 2.1, the accelerometer data can be used to estimate the trajectory of the camera and thus the PSF can be estimated. However, since we don't know the scene depth and there is a drift due to approximate value of gravity vector, we iterate through various possible depths and shifts to find the appropriate image.

4.1 Constructing the kernel

Initially, we verify that the trajectory obtained from the sensors can be used to construct a good kernel estimate. For this purpose, we took a long shot image of a single point light source. The image obtained would represent the blur kernel. Hence, we could compare the kernel we constructed and the kernel from the image. An example image and kernel pair are given in Fig.4.2.

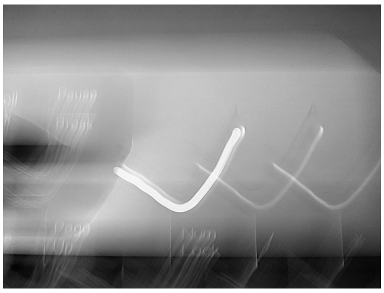

Figure 4.1: Actual image showing the point spread function, which acts as our ground truth measurement.

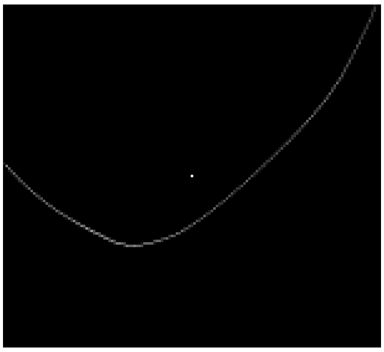

Figure 4.2: Kernel constructed from sensors data. The scale is not exact. Further, there was no drift compensation.

From the above figures, we see that the trajectory estimated from inertial sensors data gives a good initial estimate for the PSF. Note that the scale has not been calculated and the drift has not been compensated. The scale and the drift information are available during the deblurring of the image.

The above example was a good case, where there was little drift. The following example has a considerable drift, which gives rise to an erroneous kernel.

Figure 4.3: Actual image showing the point spread function, which acts as our ground truth measurement.

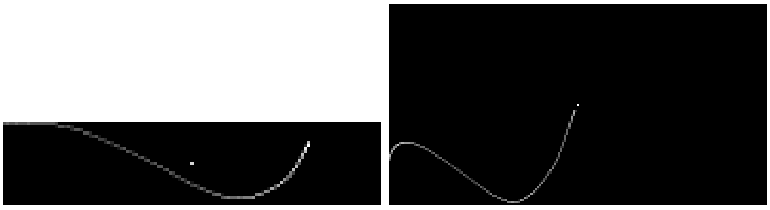

Figure 4.4: Plain kernel and the kernel corrected for drift.

Fig.4.4 shows that there is a significant drift in the trajectory, which is visible in the first constructed image. We iterated through various possible drifts and found visually that the drift correction was $(0.004284, 0.013347)mm$, i.e, the final position is offset by this amount.

4.2 Deblurring the image

Once we have the approximate kernel, we can perform deblurring. Since we don't have the scene depth, we iterate through various depths. We assumed that our kernel was very accurate and tried estimating the latent image using a simple weiner filter. Some of the results, along with blind deconvolution output by Xu and Jia (2010) are given below.

Figure 4.5: Blurred image and the best deblurred output using wiener filter.

Figure 4.6: Blind deconvolution by Xu and Jia (2010).

The method by Xu and Jia (2010) did not restore the image to a great extent. Using Wiener filter restored the image to a more sharper form but the intensity at the edges increased drastically.

As an alternate approach, we used the blind deconvolution algorithm by Abhijith Punnappurath (2014). Instead of a complete blind deconvolution, we used our kernel as an initial estimate and evaluate it against complete blind deconvolution result. We found that for highly textured images, the output was the same. However, the algorithm converged quickly when we used our kernel as an initial estimate. We also ran our algorithm for very low textured images and found that the results were better than the blind case and it took the same time to converge. The results have been given in Fig.4.9.

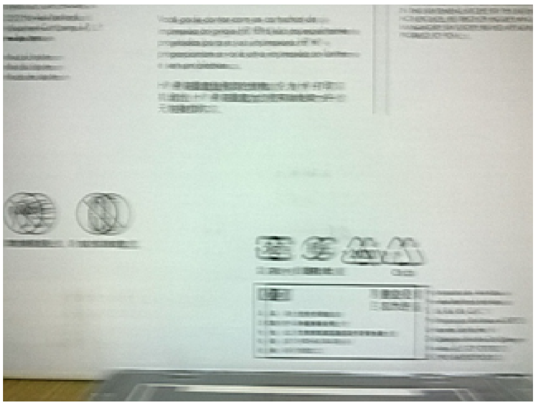

Figure 4.7: Actual blurred image.

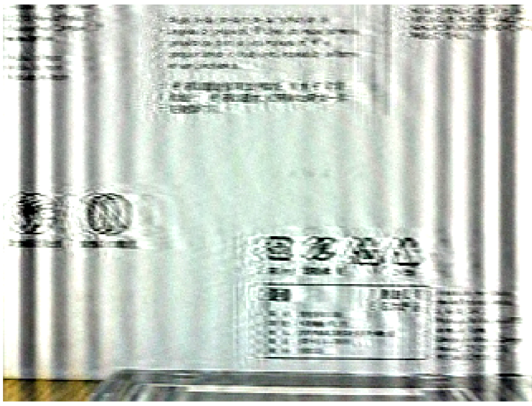

Figure 4.8: Deblurring using total blind method.

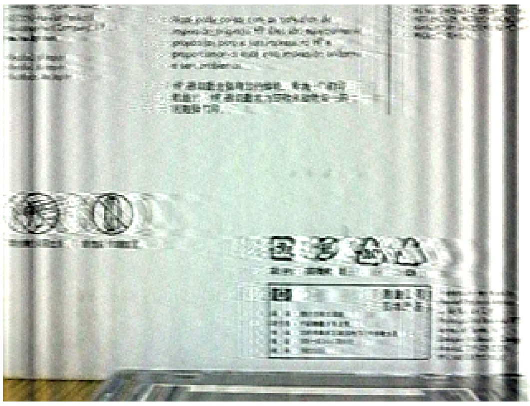

Figure 4.9: Deblurring using our kernel as the initial estimate.

As is visible from the images above, we find that the deblurring method with our kernel as the initial estimate gives better results.

CHAPTER 5

IMAGE REGISTRATION

When taking multiple images, it is very likely that the subsequent images will be shifted and rotated. Consider the case of simple shift. In the absence of noise, we could use normalized cross correlation to estimate this shift. However, this method would fail when one of the image is blurred. A more sophisticated method would use feature matching to get a good estimate of a the shift. However, the drawback is that it is computationally intensive.

To overcome this problem, we can use the inertial sensor data on the mobile to get a good estimate of the change in orientation, since the inertial sensors give direct information about the motion during the exposure time.

Before we go into the case of pure translation, we need to understand the limitation of our current setup. We have a 3-axis accelerometer giving us the acceleration along x, y and z axes, relative to the mobile frame of reference. Since our setup constrains that we have only in-plane motion, the acceleration along z axis does not give any information. Any generic motion in the plane can be represented by three entities, translation in the two axes in the plane of motion, namely t_x, t_y and rotation in the perpendicular axis, namely, R_z. Hence, we have three unknowns and only two equations from the two axes, which makes it an indeterminate system. Hence, we constrain the system by saying that either the system goes through only translation or only rotation. Hence, we have the following equations:

$$
\begin{aligned}
t_{x_i} &= \iint a_x dt \\
t_{y_i} &= \iint a_y dt \\
&or \\
\theta_i &= \tan^{-1}(a_y/a_x)
\end{aligned}
$$

We look at two cases, one of pure translation and another of pure rotation.

5.1 Pure translation

As mentioned in 3.1.1, we can determine the trajectory of the device from the accelerometer data, which is given by

$$
\begin{aligned}
x_{\mathit{final}} &= \int_{t_{init}}^{t_{final}} \int_{t_{init}}^{t_{final}} a_x dt \\
y_{\mathit{final}} &= \int_{t_{init}}^{t_{final}} \int_{t_{init}}^{t_{final}} a_y dt.
\end{aligned}
$$

We wish to find the shifts x_{shift}, y_{shift} such that the RMS difference between the first image and the shifted second image is reduced. i.e

$$
\min ||im_1(x, y) - im2(x - x_{shift}, y - y_{shift})||^2
$$

The calculated final position is in mm and needs to be converted into pixel dimension. However, we don't have the distance of the scene from the camera, which makes it an ill-posed problem. To overcome this, we iterate through various scales and find out the correct scale using RMS error. Hence,

$$
(x_{shift}, y_{shift}) = argmin_k\{\sum_x \sum_y (im_1(x, y) - im_2(x - k * x_{final}, y - k * y_{final}))^2\}
$$

Note that the maximum shift is bound due to the finite duration. Hence, our search space is drastically reduced from 360^o in the blind case to a small sector(it is not a straight line due to the drift associated with the accelerometer readings) with the help of accelerometer data.

Fig.5.2 shows example outputs of the algorithm.

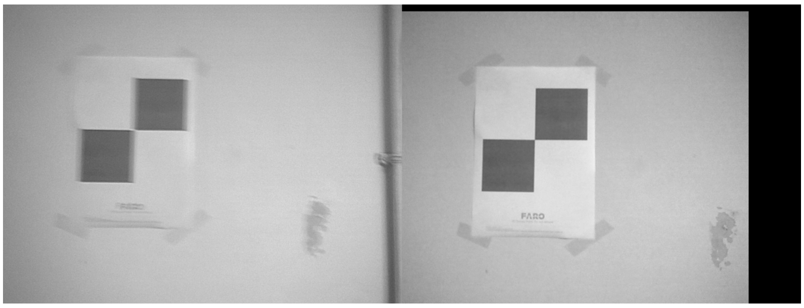

Figure 5.1: Example 1: Registered image pairs.

Figure 5.2: Example 2: Registered image pairs.

Observe that the images are registered very accurately even when one of the images is blurred. The algorithm was very fast and ran in less than 10s for both the cases. Further, as we have a good estimate of the direction of motion, finer registration can be done by increasing the search space, after performing coarse registration using the above mentioned method.

5.2 Pure rotation

Assuming only in-plane rotation, we can relate the measured acceleration to world acceleration (in the absence of other forces) as

$$a_x = g\cos\theta$$
$$a_y = g\sin\theta$$

Where θ is the orientation of the device. Hence, the orientation of the mobile can be estimated as

$$\theta = \tan^{-1}(\frac{a_y}{a_x})$$

As in the pure translation case, we take two images, one non-blurred and another blurred, both separated by 500ms duration. We only rotate the device during this duration. Hence, we can use the acceleration values estimated during this time frame to correct for the rotation. Note that the process of rotation will also induce a small translation in the image. This makes the RMS error method of finding the appropriate angle difficult. We hence use normalized cross correlation, which is independent of the shift. Thus,

$$\theta = argmax_t\{\mathcal{F}^{-1}\left(\frac{\mathcal{F}(im_1).\mathcal{F}(\mathcal{R}(im_2,\theta_t))^*}{|\mathcal{F}(im_1).\mathcal{F}(\mathcal{R}(im_2,\theta_t))|}\right)\}$$

Where $\mathcal{F}(.)$ is the 2D discrete Fourier transform, $\mathcal{F}^{-1}(.)$ is the 2D discrete inverse Fourier transform and $\mathcal{R}(im,\theta)$ rotates the image by θ degrees.

Fig.5.5 gives three examples of registration using the proposed method.

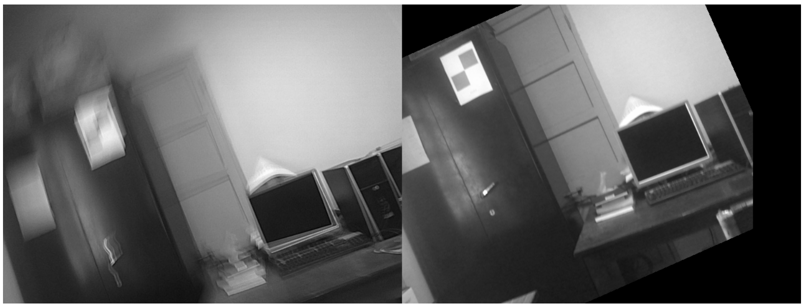

Figure 5.3: Example 1: Very high blur and rotation.

Figure 5.4: Example 2: Moderate blur and rotation.

Figure 5.5: Example 3: Very small blur and rotation.

In the first and second example, the first image, which is taken with long exposure is a highly blurred image. As is evident from the images, the algorithm performs very good. The third image is a small rotation case and our algorithm performs well for that case too.

The shift is not accounted for in the above figures. This is due to the absence of any further information from the motion sensors. The availability of a gyroscope could give the direction of shift and hence help in further registration.

CHAPTER 6

DEPTH ESTIMATION

When taking an image, due to the variable depth of the scene, the amount of blur induced at every pixel due to defocus or motion of the camera is different. We use this information to estimate the depth from images using two methods, namely depth from motion blur and depth from focus.

6.1 Depth from motion blur

Section 2.2.2 gives a brief discussion about how every point in the image gets blurred by a different blur kernel due to the depth profile. If we neglect the parallax error, we can see that the only variation in the blur kernel is the scale. Hence, estimating the depth profile boils down to estimating the scale of the blur kernel at different points. Estimating depth from a blurred and latent image has been discussed in Paramanand and Rajagopalan (2010). However, the method requires estimation of depth at every point, due to the absence of any motion information. Levin *et al.* (2007*b*) uses coded aperture for extracting the depth profile from the image. However, the paper has good discussion about estimating depth from two images, which is more apt for our discussion.

In the experimental setup, we capture two images, one with very short exposure and another with long exposure. Further, along with the two images, the motion of the camera is estimated by logging the accelerometer values. Thus, we have the latent image, the blurred image and the blur kernel. We need to estimate the scale of the blur kernel at each point. Prior to estimating the depth, it is very important that the two images are aligned. We would show that misalignment could be a major reason for wrong depth profiles. However, for small shifts, we found that iterating through a small space to search for the right shifts was enough.

To estimate the depth at each point, we iterate through various blur kernel sizes and calculate the difference between the blurred image and the blurred latent image for each scale of the kernel. If the scale is correct at a given position, the error at that point will be the least and that would be assigned as the depth value. Hence, we can formulate our algorithm as

$$k(x, y) = argmin_d\{|im(x, y)_{blurred} - h(x, y, d) * im_{latent}(x, y)|^2\}$$

Where $h(x, y, d)$ is the kernel with scale value of d. While this works for the ideal case, the real depth map is very noisy. Hence, we low pass filter the error map with a box filter. This algorithm works very well when there is good texture in the image. Further, it is necessary that the kernel is very accurate, else the depth map will go wrong.

To verify our algorithm, we first tried estimating depth map in a synthetic case. We blurred an image with a known depth map and ran our algorithm. The results in Fig.6.2 were obtained.

Figure 6.1: Latent image and synthetic lurred image pair.

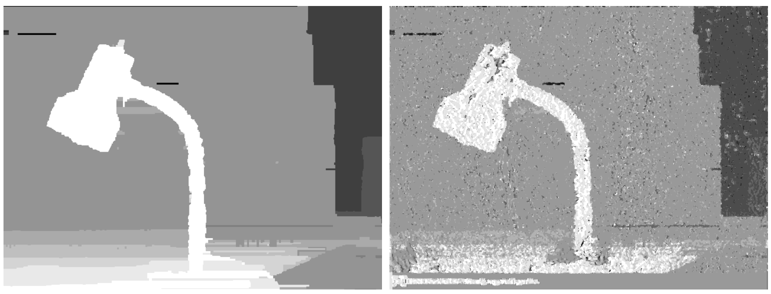

Figure 6.2: True depth map and estimated depth map.

The obtained depth map was close to the ground truth depth map. We then tested our algorithm for real cases. In our experiment, we found that the kernel can go to a maximum size of 20×20 and hence we scaled the kernel size from 1×1 to 20×20. We used a filter of size 32×32 for the error map. Some of the experimental results have been in Fig.6.6.

Figure 6.3: Latent image and Blurred image pair. This is an example of close up shot.

Figure 6.4: Estimated depth map. Brighter area represents scene closer to the camera.

Observe that the image pair are very well aligned already and the object is well textured. This made estimation of depth map accurate. However, the following example is a failure case for estimating depth.

Figure 6.5: Latent image and Blurred image pair. Note that there is a misalignment between the images.

Figure 6.6: Bad depth map. While the depth calculation at the edges is expected to be accurate, the obtained depth map has erroneous edge values.

To find out the reason for a bad depth map, we perturbed the kernel while estimating depth in synthetic case. It was found that if the images are misaligned or if the kernel is not accurate, the output depth map was not accurate.

6.2 Depth from focus

Estimating depth from focus is termed as Structure From Focus (SFF). As discussed in 2.2.1, varying a parameter of the camera system changes the quality of the image at different depth. We are interested in varying the focus of the camera to vary the *sharpness* of the image. The sharpness of the image is measured using the focus operator. A very popular and easily implementable focus operator is the Sum Modified Laplacian (SML). The Modified Laplacian operator is defined as

$$\Delta_m I = |I * G_x| + |I * G_y|$$
$$G_x = [\begin{array}{ccc} -1 & 2 & -1 \end{array}], G_y = G_x^T$$

The Sum Modified Laplacian operator does a average of the Modified Laplacian over the neighbors. Thus,

$$\phi(x, y) = \sum_{i=-N}^{N} \sum_{j=-N}^{N} \Delta_m I(x - i, y - i)$$

When a region is in focus, the SML value is the highest. The experiment consists of taking 100 images with focus varying from minimum to maximum. For each image, the SML operator is applied and the appropriate depth for each pixel is when the focus operation peaks.

When changing the focus value, there is a scaling in the images. The robust way to correct this would be to register each image with the first image using Fourier Mellin Transform or feature based image registration. However, we found that finding the scale of the final image with respect to the first image and interpolating the scale for all other images worked good. Further, this scaling was consistent across all the experiments and hence finding the necessary scaling was a one time operation.

The following image pair is the first and the last image in the 100 images.

Figure 6.7: First and last image of the 100 images. Note that in the first image, the screw driver handle is in complete focus and in the last image, the book is in complete focus.

To evaluate our algorithm, we used a filter size of 32×32. Further, to confirm that our algorithm returns the correct depth map, we created an all focus image from the set of images. Fig.6.9 shows the obtained results.

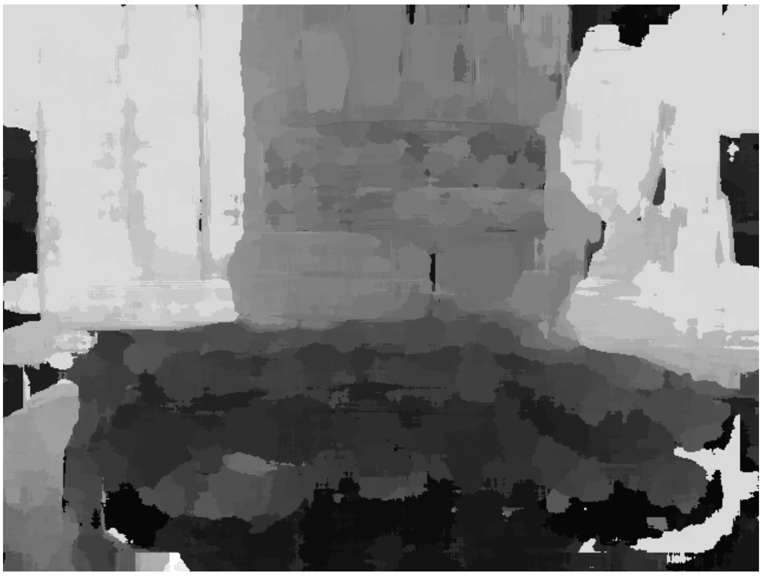

Figure 6.8: Estimated depth map. Objects nearer to the camera are darker. Observe how the objects have been segregated.

Figure 6.9: All focused image.

From the depth map, we see that the objects have been segregated on the basis of depth. Further, the areas of low texture don't respond very well, as the SML operator is a high frequency measurement operator.

CHAPTER 7

CONCLUSION

We have evaluated some of the important image restoration and registration algorithm with data from a mobile. Some of the results are very promising, while some of them weren't very good. In particular, estimating depth from focus, and image registration have given very good results. Depth estimation from blurred and latent image pair was good to a certain extent but was very sensitive to the accuracy of the kernel estimate. The results can be made better with the presence of a magnetometer, which would provide accurate orientation information, thus making the estimate of trajectory of motion more accurate. The image deblurring, while proving interesting results like fast convergence and better results in case of low texture images, has good scope for improvement.

While all the codes were run offline for ease of research purpose, we saw that the algorithms were very simple and efficient due to availability of information from motion sensors. Hence, porting these algorithms onto a mobile should be an easy task.

7.1 Future work

Along with the evaluation of major algorithms for registration and restoration, the project has provided a good stage for future development. All the discussed algorithms can be immediately ported to a mobile platform with great ease, which would provide a very powerful computational imaging device.

Along from the usefulness of our algorithms, the mobile application is a very versatile one and would serve well for future experiments with different algorithms. With features like continuous logging of images, continuous logging of motion sensor data, capturing low exposure noisy image and long exposure blurred image and freely changing parameters of the camera, the device would prove very useful for working on computational imaging on a mobile platform.

Apart from the sunny side of the project, there were some parts which were not addressed appropriately. Not much effort was put into writing a more efficient semi-blind deconvolution algorithm. However, this was partly not possible due to absence of magnetometer on our device. Further, the mobile currently does all the job using a 10ms timer, which might be inefficient. Unloading the work onto a separate thread will not only make it neat, but it would make it much more efficient.

We also started working on multi-frame super-resolution but were not able to go further due to time constraints. As registration forms a major part of super-resolution, the data from the inertial sensors can greatly aide in reducing the complexity of this step.

We are looking forward to open our code to public so that people who are interested can contribute. The code will be the property of the lab and the author of the report would like to credit all the work to the lab.

REFERENCES

Abhijith Punnappurath, G. S., A. N. Rajagopalan, Blind restoration of aerial imagery degraded by spatially varying motion blur. *In Society of Photographic Instrumentation Engineers, 2014. Proceedings SPIE'14*. 2014.

Bae, H., **C. C. Fowlkes**, and **P. H. Chou**, Accurate motion deblurring using camera motion tracking and scene depth. *In WACV*. 2013.

Bioucas-Dias, J. M., **M. A. Figueiredo**, and **J. P. Oliveira**, Total variation-based image deconvolution: a majorization-minimization approach. *In Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, volume 2. IEEE, 2006.

Chan, T. F. and **C.-K. Wong** (1998). Total variation blind deconvolution. *Image Processing, IEEE Transactions on*, **7**(3), 370–375.

Fergus, R., **B. Singh**, **A. Hertzmann**, **S. T. Roweis**, and **W. T. Freeman**, Removing camera shake from a single photograph. *In ACM Transactions on Graphics (TOG)*, volume 25. ACM, 2006.

Freeman, W. T. and **E. H. Adelson** (1991). The design and use of steerable filters. *IEEE Transactions on Pattern analysis and machine intelligence*, **13**(9), 891–906.

Gupta, A., **N. Joshi**, **C. L. Zitnick**, **M. Cohen**, and **B. Curless**, Single image deblurring using motion density functions. *In Computer Vision–ECCV 2010*. Springer, 2010, 171–184.

Jin, H. and **P. Favaro**, A variational approach to shape from defocus. *In Computer VisionâĂŤECCV 2002*. Springer, 2002, 18–30.

Joshi, N., **S. B. Kang**, **C. L. Zitnick**, and **R. Szeliski** (2010). Image deblurring using inertial measurement sensors. *ACM Transactions on Graphics (TOG)*, **29**(4), 30.

Jung, S.-H. and **C. J. Taylor**, Camera trajectory estimation using inertial sensor measurements and structure from motion results. *In Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 2. IEEE, 2001.

Krishnan, D. and **R. Fergus**, Fast image deconvolution using hyper-laplacian priors. *In NIPS*, volume 22. 2009.

Levin, A., **R. Fergus**, **F. Durand**, and **W. T. Freeman** (2007*a*). Deconvolution using natural image priors. *ACM Transactions on Graphics (TOG)*, **26**(3), 0–2.

Levin, A., **R. Fergus**, **F. Durand**, and **W. T. Freeman** (2007*b*). Image and depth from a conventional camera with a coded aperture. *ACM Transactions on Graphics (TOG)*, **26**(3), 70.

Long, Y.-W. T. P. T. and **G. M. S. Brown** (). Richardson-lucy deblurring for scenes under projective motion path.

Losch, S. (2009). *Depth from Blur: Combining Image Deblurring and Depth Estimation*. Ph.D. thesis, Department of Computer Science, Saarland University, SaarbrÃijcken, Germany.

Money, J. H. and **S. H. Kang** (2006). Total variation semi-blind deconvolution using shock filters. *submitted to Image and Vision Computing*.

Network, M. D. (2013–). Windows phone samples: learn through code. URL `http://code.msdn.microsoft.com/wpapps/`.

Ojansivu, V. and **J. Heikkila** (2007). Image registration using blur-invariant phase correlation. *Signal Processing Letters, IEEE*, **14**(7), 449–452.

Oliveira, J. P., **M. A. Figueiredo**, and **J. M. Bioucas-Dias**, Blind estimation of motion blur parameters for image deconvolution. *In Pattern Recognition and Image Analysis*. Springer, 2007, 604–611.

Paramanand, C. and **A. Rajagopalan**, Unscented transformation for depth from motion-blur in videos. *In Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*. IEEE, 2010.

Pertuz, S., **D. Puig**, and **M. A. Garcia** (2013). Analysis of focus measure operators for shape-from-focus. *Pattern Recognition*, **46**(5), 1415–1432.

Portilla, J. and **R. Navarro**, Efficient method for space-variant low-pass filtering. *In VII National Symposium on Pattern Recognition and Image Analysis*, volume 1. 1997.

Reddy, D., **J. Bai**, and **R. Ramamoorthi** (). External mask based depth and light field camera.

Šindelář, O. and **F. Šroubek** (2013). Image deblurring in smartphone devices using built-in inertial measurement sensors. *Journal of Electronic Imaging*, **22**(1), 011003–011003.

Sroubek, F. and **P. Milanfar** (2012). Robust multichannel blind deconvolution via fast alternating minimization. *Image Processing, IEEE Transactions on*, **21**(4), 1687–1700.

Subbarao, M. and **N. Gurumoorthy**, Depth recovery from blurred edges. *In Computer Vision and Pattern Recognition, 1988. Proceedings CVPR'88., Computer Society Conference on*. IEEE, 1988.

Vandewalle, P. (2006). *Super-resolution from unregistered aliased images*. Ph.D. thesis, Ecole Polytechnique Fédérale de Lausanne.

Wang, C., **L. Sun**, **Z. Chen**, **S. Yang**, and **J. Zhang**, Robust inter-scale non-blind image motion deblurring. *In Image Processing (ICIP), 2009 16th IEEE International Conference on*. IEEE, 2009.

Xiao, F., **A. Silverstein**, and **J. Farrell**, Camera-motion and effective spatial resolution. *In International Congress of Imaging Science, Rochester, NY*. 2006.

Xiong, Y. and **S. A. Shafer**, Depth from focusing and defocusing. *In Computer Vision and Pattern Recognition, 1993. Proceedings CVPR'93., 1993 IEEE Computer Society Conference on*. IEEE, 1993.

Xu, L. and **J. Jia**, Two-phase kernel estimation for robust motion deblurring. *In Computer Vision–ECCV 2010*. Springer, 2010, 157–170.