Predictive filtering for automatic focus pulling and robot control

Ehsan Kia

Master of Science

School of Computer Science

McGill University

Montreal, Quebec

December 2016

A thesis submitted to McGill University in partial fulfillment
of the requirements of the degree of master of science.

ACKNOWLEDGEMENTS

The author wishes to express his deepest appreciation to his supervisor, Paul Kry, who provided unwavering guidance and assistance. Special thanks to all the members of the computer graphics lab for the support, discussions and ideas.

ABSTRACT

We present a system for automatic focus pulling using consumer level hardware and predictive Kalman filters. This work describes the complete process of building such a system from scratch, both in hardware and software. We document how to modify a Canon lens to communicate with it directly, and how to communicate with it using a micro-controller. We then calibrate the lens focus motor and measure the latency of the system. We next introduce a series of predictive filters aimed at a variety of focus targets that could arise in a film. These filters compensate for the latency in the system and aim to keep the target in perfect focus while in motion. Finally, we test each of these filters using the modified lens to track the predicted position of the target. We evaluate the defocus in the examples both synthetically from the recorded motion data and numerically by analyzing the recorded video for blur. We also showcase a few other uses of our predictive Kalman filters in use with quadrotors.

ABRÉGÉ

Nous présentons un système de mise au point automatique tirant à l'aide du matériel de niveau des consommateurs et des filtres de Kalman prédictifs. Ce travail décrit le processus complet de la construction d'un tel système à partir de zéro, tant au niveau matériel et logiciel. Nous documentons comment modifier un objectif Canon pour communiquer directement avec elle, et la façon de communiquer avec l'aide d'un micro-contrôleur. Nous calibrons alors le moteur de mise au point de l'objectif et nous mesurons la latence du système. Nous présentons ensuite une série de filtres prédictifs visant une variété de cibles de mise au point qui pourraient survenir dans un film. Ces filtres compensent la latence dans le système et visent à maintenir la cible au point parfaite, tout en mouvement. Enfin, nous testons chacun de ces filtres en utilisant l'objectif modifiée pour suivre la position prédite de la cible. Nous évaluons la défocalisation dans les exemples à la fois synthétiquement à partir des données de mouvement enregistrées et numériquement en analysant la vidéo enregistrée pour le flou. Nous présentons également quelques autres utilisations de nos filtres de Kalman prédictifs en utilisation avec des quadrotors.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
Introduction

One of the most challenging jobs in filmmaking is that of the first camera assistant who must control the lens so that actors or objects are in focus throughout a shot (Figure 1–1). Positions and distances can be established in rehearsal, but *pulling focus*, as it is known, is difficult when the target is in motion. Additionally, there exist a variety of scenarios where shallow depth of field makes this task even more challenging. In particular, the depth of field becomes very shallow when using a wide open aperture, which may be desirable to produce a dramatic blurry background or to direct the viewer's attention to a specific point in the image. In this case, the depth of field can be as small as a centimeter, especially when using medium length or telephoto lenses, but also when filming at close distances with a wide angle lens. A small error can be the difference between focusing on the actor's ears instead of their eyes.

An interesting solution for difficult focus pulling scenarios is to use a motion capture system to measure the location of actors and objects, and to drive the camera focus automatically with a motor. While such an approach can trivially produce sharp focus when everything is static, the end to end delay from measurement to control of the focal-plane will result in a soft defocus when either the camera or target is in motion. This latency exists in all measurement systems. For instance, magnetic tracking systems, while ideal for this application because the sensors can be hidden on actors and objects, typically contribute at least 15 ms latency. Furthermore, magnetic tracking measurements have noise that requires filtering, which introduces additional latency due to the phase shift. Data processing, communication, and motor control are additional sources of delay.

Figure 1–1: The first camera assistant takes care of focus pulling and makes sure the focus always follows the target as it moves around the scene. *Photograph by Clemens Pfeiffer, distributed under a CC-BY 2.0 license*

In this work, we use predictive filtering to address the question of how one can overcome end to end latency and automatically maintain focus on moving targets. We design predictive (delay compensation) filters for focus pulling in such systems with inherent delay. In some cases, we can expect intermittent defocus whenever the motion of an object deviates from an assumed motion model. For instance, consider an object moving at a constant velocity that suddenly stops due to an inelastic collision. We can predict the position from past measurements, but once the object stops, the error in our prediction will grow linearly until we obtain measurements that show that the object is at rest.

We evaluate video recorded with the camera shown in Figure 1–2 using a motion capture system to control the focus. We use two different electro-focus lenses that we modified to obtain full and direct control of the focus and aperture. We carefully measure the end to end latency and combine this with motion models for

camera and targets to decide when and where to set the focal plane. Specifically, a Kalman filter with a variety of different process models estimates the state of the camera, people, and objects (e.g., position, velocity, acceleration). It then integrates the model forward in time from the state estimate which compensates for the latency. We set measurement and model noise parameters using values optimized from example data.

We explore a wide variety of different models, designing and evaluating predictive filters for each. There exist a number of scenarios where motion can be predicted with reasonable accuracy over small time windows. A constant velocity model works well for tracking people and handheld cameras. Objects that are sliding, thrown or falling are modeled well with constant acceleration. We also present examples that involve non-smooth dynamics, such as collision and static friction. Indeed, our predictions will not be perfect due to modeling errors and measurement noise, and thus, we evaluate the quality of results for a range of examples.

Direct measurement of blur in each frame of a video can be used to evaluate results, but in general, we compare future motion capture measurements with previously predicted focal plane settings. From this, we can estimate the blur by taking into account the camera model, lens focal length, and distances to both the focal plane and the target. In a concurrent research project by our lab, a set of perceptual defocus experiments were performed on subjects and a momentary blur threshold was identified. This helps us detect cases where a certain defocus in a video becomes problematic, given its blur amount and duration. For instance, end to end latency can become particularly challenging when recording high-speed video as short intervals of defocus will be longer at playback time and become noticeable. Our threshold results are also useful in guiding aperture selection so as to control the depth of field, ensuring that any temporary defocus remains below the threshold.

1.1 Thesis organization

In this work, we present a complete system built from the ground up that allows us design, record and evaluate predictive filters for automatic focus pulling. We describe the process of modifying and calibrating a consumer lens to allow precise control over the focus motor. Then, we measure the system and use those measurements to design and tune predictive filters which compensate for the latency measured. Finally, we evaluate how our filters perform in a wide variety of examples.

The thesis is organized as follows and will describe in depth each part of our process. In Chapter 2, we will give a detailed overview of the related works with references for every section of our work. Then, in Chapter 3, we will take a close look at modifying consumer level lenses and cameras to get a precise control programmatically. Next, we explore how to design and optimize filters to reduce blur in different scenarios in Chapter 4. Lastly, in Chapter 5, we evaluate our system and filters using a variety of different examples and evaluation methods.

Figure 1–2: The camera and one of the modified lenses we use for recording videos with automatic focus pull using predictive filtering. Blue wires attached to the lens allow direct control of the lens from a computer using a serial communication protocol. Four small retro-reflective markers mounted on top of the camera allow its position and orientation to be tracked with a motion capture system.

CHAPTER 2
Related work

The presence and control of depth of field blur are important in computer generated images and photography. Demers [2004] provides a survey of rendering methods for depth of field, while more recent work has addressed real-time techniques [Lee et al., 2010] and accuracy with efficient sampling [Belcour et al., 2013]. In computer vision, defocus can be used to estimate depth [Subbarao and Surya, 1994], and can aid in scanning [Jakob et al., 2009]. Blur also has a strong effect on a viewer's perception of distances in an image. Blur gradients provide perceptual cues about scene scale [Held et al., 2010], and blur at occlusion boundaries can provide information about depth order [Mather and Smith, 2002]. Depth of field blur can also improve perception in renderings of volumetric data [Grosset et al., 2013]. In our work, we are concerned with the amount of blur when viewing recorded video on a screen. With an understanding of how people see defocus in objects moving in depth, we can design and evaluate methods for reducing blur in video recorded with focus controlled by motion capture. The ultimate goal is to design predictive filters that are optimized based on human perception of depth as as seen in video. Below, we discuss related work on perception, focusing, and filtering.

2.1 Blur perception

There have been many studies that address this question in the perception literature. Most of these studies examine static stimuli only. A well-known finding is that blur discrimination thresholds at large reference blurs obey roughly a Weber law, so just noticeable differences (JNDs) in blur are proportional to a reference blur level. At small blur references (up to around 2 arcmin), blur discrimination thresholds exhibit a dipper function. This dipper function exists both in the fovea [Watson

and Ahumada, 2011] and in the periphery [Wang and Ciuffreda, 2005]. The reason for the dipper function is that, at a zero reference blur, the discrimination threshold becomes equivalent to the detection threshold which is limited by factors such as photoreceptor sampling, noise, and optics. The visual system thus only can detect blur differences if the blur is above some minimum blur detection level, which is roughly where the dipper has its dip. This detection limit also roughly corresponds to the depth of field limit of blur in 3D scenes.

Focus pulling involves objects that move in depth. The goal is to track the motion in depth so that it does not pass outside the camera's depth of field. Although lateral motion does elevate blur discrimination thresholds, the elevation was found to be small even for large 2D image velocities [Burr and Morgan, 1997]. For the problem of focus pulling, the motion is often primarily due to objects either approaching or receding in depth and the lateral component of motion is typically low. In this case, motion blur is likely a negligible factor for blur detection and discrimination. The more important factor is likely to be errors in focus pulling. Such errors might result in blur that can have a constant component (if there is a fixed delay in the tracking of motion in depth) or a time-varying component, for example, if the tracking mechanism overshoots or undershoots.

2.2 Camera modification

In the field of computational photography, there have been various projects trying to provide lower level access to the different components in a camera. One such example is the Frankencamera from Stanford [Adams et al., 2010], which specifies a software stack and a set of APIs for precision access to the camera's sensor and processing pipeline. In a similar vein, there has been work on computational cameras which explore the benefits and limits of using computation with novel optics to produce better images [Zhou and Nayar, 2011], as well as research on the convergence of optics and processing [Nayar, 2011]. Another project that we investigated

is Magic Lantern, a project dedicated to writing custom firmware for Canon cameras. While they provide many useful new functionalities, their focus control was still not precise enough for us.

As for modifying consumer level lenses, there have been a number of attempts documented on the web without always having a corresponding academic publication. Two such references which helped in reverse engineering the communication protocol were Jean Wlodarski's Pick and Place blog [Wlodarski, 2011], as well as a supplementary document posted by Yosuke Bando alongside his research [Bando et al., 2013]. It is worth noting that these communication protocols are often lens dependent and often incomplete, as most of the posted results were focused on specific use cases.

2.3 Autofocus and re-focus

Most consumer photography cameras use phase detection in their autofocus systems. The autofocus sensor compares the light patterns coming from opposite sides of the lens. When the patterns do not match, the shift amount tells exactly how much to move the lens and in what direction. Modern cameras use a collection of these sensors and provide modes for guessing which point of the image should be in focus. For instance, the Canon 70D provides an autofocus mode that can track faces when recording video.

Light field photography is a potentially interesting solution to the focus tracking problem because the light can be refocused in a post-processing step [Ng, 2005]. The Lytro Illum camera, which is based on Ng's work, provides excellent quality images that are refocusable. In the past, bandwidth for video recording was a big challenge given that the light fields are recorded with a 40 megapixel (mega-ray) image sensor, but their most recent camera, the Lytro Cinema, overcomes this issue. They have built a light field camera able to record videos, although the technology is still very expensive and only available to big budget labs or movie studios.

Other recent work includes camera arrays that are small enough to be built into cell phones [Venkataraman et al., 2013]. Raytrix makes light field cameras targeted to industrial applications, and has a video capable camera, though the resolution and quality are probably insufficient for most entertainment applications.

The assistant camera operator, or AC, typically tracks focus using a handheld remote. Cinematography Electronics produces the Cine Tape Measure. It uses an ultrasonic sensor to continuously measure the distance to the subject. For automatic focus at longer distances, there exist optical infrared laser distance measurement solutions (e.g., cmotion and cfinder). Moviecam Easyfocus also uses optical measurement and allows the camera assistant to click on a touch screen to select the subject. The Andra autofocus systems, recently available from Cinema Control Labs, gives the AC high-level tools for controlling focus on actors and objects that are tracked with motion capture.

Our autofocus implementation is inspired by the Andra system, though we use an optical motion capture system instead of wireless magnetic sensors. A number of other commercial solutions exist. These professional systems have low latency but are also very expensive.

2.4 Predictive filtering

Predicting motion is a critical problem across many domains. It is not only the problem of optimal state estimation but also the prediction of present and future states given past measurements. For interactive techniques, latency can be the primary challenge. To improve the quality of augmented reality, Azuma and Bishop [1994] describe how to compensate for delay in head mounted displays. Prediction of mouse motion has been studied for navigation in distributed virtual environments [Chan et al., 2005], for telepresence applications [Baldwin et al., 1998], and for predicting mouse endpoint positions in human-computer interaction [Pasqual and Wobbrock, 2014].

Simon [2006] provides an excellent background on optimal state estimation with Kalman filters. Estimator performance can be poor when measurements are not only delayed but also have uncertain time stamps [Julier and Uhlmann, 2005]. There are also interesting problems when the measurements have variable delays due to the speed of light and sound [Orguner and Gustafsson, 2008]. In our case, the problem of tracking moving targets in a studio sized environment with a fixed end-to-end latency is much simpler, and our approach closely resembles the two-step process of Azuma and Bishop. We first use a simple steady state Kalman filter, also known as an α-β or α-β-γ filter, to estimate the past state of a target. We then integrate forward a model (possibly non-smooth) to predict where to set the focus.

One relevant domain where predictive filtering shows up often is robotics. Messom et al. explore the usage of Kalman filtering in improving control of a mobile robot [Messom et al., 2003]. They explore techniques to improve the precision of their soccer playing robot using predictive filter and examine the results on the field. In a similar vein, Kiruluta et al. use similar predictive Kalman filters on tracking head movements [Kiruluta et al., 1997]. The textbook by Liu [2009] is also a great reference on the subject.

2.5 Summary

This project pulls together a wide variety of domains, touching both software and hardware. While there are many related works in different fields, none of the research tackles the problem we are trying to solve here. There are a few commercial solutions which attempt to solve the focus tracking problem, but they are generally very technologically complex and particularly expensive. We propose a solution which uses predictive filtering to achieve acceptable results using cheap and accessible consumer grade equipment. We also expand on the vast amount of ad-hoc reverse engineering work that has been done in various digital photography

communities online. Lastly, we bring in research from the psychophysics domain in order to evaluate our results and confirm that we are below perceivable thresholds.

CHAPTER 3
Focus control

Professional filmmaking involves expensive cameras and lenses. Typically, external lens motors of various brands are used to drive the focus ring. The assistant camera operator uses a handheld remote to adjust the focus, and this equipment normally has reasonably low latency for good manual control. While we could implement predictive filtering for these traditional systems, we build an inexpensive alternative using consumer cameras and lenses designed for photography.

The high quality of video and low prices of recent DSLR cameras have made them very attractive equipment for amateur filmmakers. We note that the main drawback of using photography lenses for video is that they tend to *breath* during focus, i.e., there is a small zoom during focus that changes the framing of the image. Nevertheless, the advantage is that very good optics can be obtained at affordable prices. For our work, these lenses also provide an excellent solution for focus control because they already contain motors for the camera's autofocus system. By using the motors inside the lens, we avoid the problem of trying to drive the focus with external motors, which is usually difficult or impossible on many photography lenses because they use a clutch system as opposed to providing direct mechanical control with hard stops.

3.1 Lens modification

We use Canon cameras and Canon electro-focus (EF) lenses. While EF lenses provide a great inexpensive focus control solution, the standard Canon SDK interface provides a limited access for controlling the focus. It only allows us to take preset size steps and the commands have unpredictable delays. The Canon custom

Figure 3–1: Canon EF lens modification, bypassing the connection between the lens and the camera inside the lens.

firmware project Magic Lantern also suffers from similar limitations. To have precise and interactive control of the focus motor, we modify the lens to bypass the camera and communicate with the lens directly. We disconnect the internal lens circuitry from the connectors on the lens mount, and pass communication wires through the side of the lens body (see Figure 3–1). To make the camera operate as if a manual lens is attached, it is also necessary to insulate the connectors on the lens mount. We modified two lenses: a wide angle 28 mm lens, and a medium telephoto 85 mm lens.

3.2 Serial communication

With the communication wires exposed, we use an Arduino board[1] to control the lens, to which we send commands from a computer via the USB connection.

[1] An open-source microcontroller (`https://www.arduino.cc`)

This follows closely the approach used by Bando et al. [2013], though we found it necessary to use an 84 MHz Arduino Due and carefully written signal timing code to successfully communicate using a 500 kHz clock cycle, as was necessary for our 85 mm lens (see Appendix A for timing details).

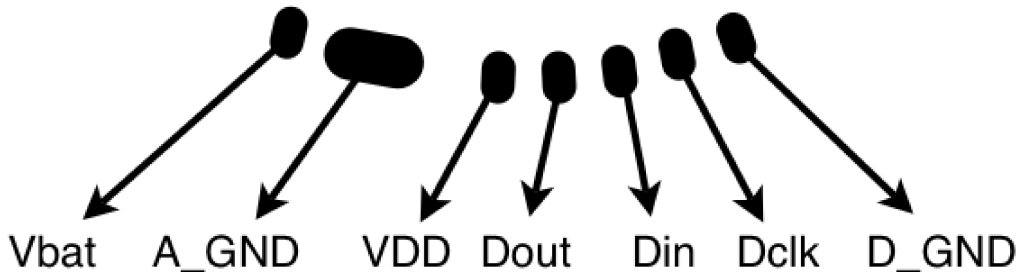


Figure 3–2: The pin configuration for the Canon EF lens. [Wlodarski, 2011]

There are 8 pins on the lens which we need to connect to in order to control it, as seen in Figure 3–2. From left to right, the first two are the power and ground pins for the motor. We provide a 6V supply, although others have had success with lower voltages too. The lens can communicate without these pins, but they are needed for moving the focus and aperture. The next pin, *VDD*, is the logic voltage, which we provide 5V using the Arduino. The next two are the data output and input pins, which are used to communicate with the lens, followed by the clock pin. Lastly, there is the logic ground pin.

The Canon protocol used by the lens resembles a normal SPI protocol[2] , but with a slight twist. After sending a command, the camera frees the clock pin and waits for the lens to pull the line high, telling the camera that it is done performing

[2] Serial Peripheral Interface, a serial communication specification primarily used in embedded systems

the command. For this reason, we had to implement our own custom communication interface.

For controlling the lens, the list of commands that were reverse engineered are shown in Table 3–1. Ultimately, we only use a small set of those to control the aperture and focus motor position. Also note that these are specific to the lenses we were using, and some of the commands are different from what was found by the other projects we used as a reference.

These motor commands on the lens all work relative to some reference start position. When the lens is first powered on, that motor current position is assigned to 0, and any further command will move relative to that. For this reason, before starting, we first pull the focus all the way back and reset the lens in order to have the 0 at the minimum focus distance. From there, we can query the lens for the motor position and know exactly where the focus is, using the calibration function detailed in the next section. The Arduino script used for communicating with the lens can be found in Appendix A.

We find that the ultrasonic stepper motors in our lenses can be reliably controlled, with no observable backlash nor any skipped steps during extended periods of use.

3.3 Focus calibration

To control the focus of our lenses, we need a mapping between the position of the stepper motor, or *ticks*, and the resulting focal distance. Rather than trying to accurately model the optics of a commercial compound lens – a daunting task – we use a simple linear relationship between ticks and inverse focal distance (diopters). This simple model works very well for our lenses (e.g., see Figure 3–3). We fit the linear function using samples collected for different focal planes measured to a fixed landmark on the camera (i.e., distance to the camera body reference frame). But to get a *best* linear fit we must adjust how we measure the distance from the camera

Command (hex)	Lens Answer	Description
0x97, 0x01	7 bytes	Ask for lens info, such as Lens ID, brand, protocol version, min/max zoom, etc
0xB0	4 bytes	Ask for max/min aperture values
0xA0	2 bytes	Ask for current zoom value
0x0A	1 bytes	Check if camera is busy (busy poll). The camera responds with 0xAA whenever it is ready
0xC0	2 bytes	Get the relative position of the focus motor in ticks (since the last reset)
0x44, 0x??, 0x??		Move the focus motor relative to its current position. The number of ticks is specified by the last 2 bytes sent (shown as 0x?? 0x??)
0x05		Set the focus motor to the furthest position
0x06		Set the focus motor to the closest position
0x13, 0x80		Set the aperture size to the minimum value

Table 3–1: A list of camera commands, the lens response and their description. These commads can vary slightly between different lens models

depending on the lens that is attached. Specifically, we find the offset that yields the smallest residual after a linear fit. The calibration was then tested in both directions to check for any backlash in the focus motor. Together, the linear function and offset allow us to compute the desired tick count by linearly interpolating between the collected data points, given a target focal plane distance measured with respect to the camera frame.

To have the focus follow a fast moving target it is important to know that the lens motor will be capable of changing the focal plane fast enough. For the 28 mm lens, the motor speed is approximately 3800 ticks per second (the full range is 1800 ticks) or about 12 diopters per second. Thus, the maximum speed of a target is about 2 m/s when at the closest focusable distance. The motor and the full range of ticks in the 85 mm lens is very similar, but given the different optics, the focus rate is about 3.6 diopters per second making the maximum speed 1.5 m/s at the closest

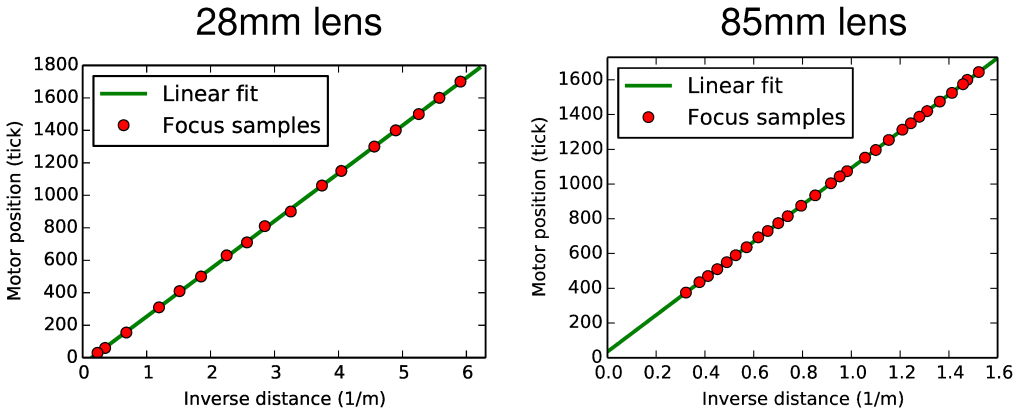


Figure 3–3: Calibration points and linear fit for 28mm and 85mm lenses.

focusable distance. In general, the maximum speed is rd^2 at distance d where r is the rate at which the lens can change focus in diopters per second. Given how quickly the speed limit increases with the square of the distance, we rarely need to worry about the ability of our lenses to keep up with fast moving targets.

These calibrations are done for focus at the center of the lens. For most common lenses, the difference when trying to focus on a subject on the edge of the frame is minimal and should not have a noticeable impact, but in the case of more complex ones such as anamorphic or tilt-shift lenses, special calibration may be needed. In such cases, we would need to also take into consideration how the focus changes as the subject moves laterally, and take that into account when computing the motor position needed to keep the target in focus.

3.4 End to end latency

Different capture systems have different amounts of latency. A magnetic tracker, such as a Polhemus, is most suitable for this application because the markers can be hidden on actors or in props. However, to reduce noise, the Polhemus system typically filters data even before it is given to the user. This involves a soft low-pass filter for when objects are moving, with little additional phase delay introduced by the filter, but it also dynamically switches to a more aggressive low-pass filter when

low speeds are detected. In our application, it is best to avoid filtering at the level of the measurement system to permit better estimates of measurement noise. It is likewise desirable to have fixed latency properties of the measured position to simplify the delay compensation problem. In our work, we use an inexpensive OptiTrack motion capture system. This typically leaves visual markers in the video but allows us to make simple tests with available equipment.

Our end to end system has many sources of latency: communication from motion capture cameras to computer, 3D reconstruction and tracking, communication to a focus driving process, USB communication of commands to an Arduino, serial communication with the lens, which then drives the stepper motor.

To measure the end-to-end latency in the system, we use a 1200 fps high-speed camera to record both the lens motion and a tracked object as it comes to an abrupt stop. In the setup shown in Figure 3–4, we hit a block of wood on a table, while both the table and the lens are in view. The end to end latency is then easily measured by counting the number of frames from when the block hits the table and when the lens' motor stops turning in the slow motion video.

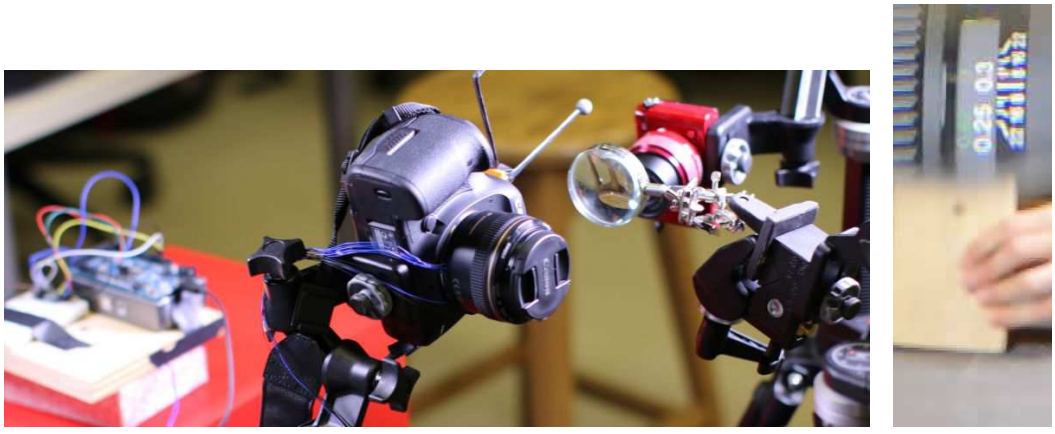


Figure 3–4: Setup for measuring the latency of the system. Left, the configuration of the lens and high-speed camera. Right, the actual view from the high-speed camera, seeing both the lens and the block hitting the table.

We measured two configurations. One configuration has the focus driving process on another computer as the motion capture machine, while the other avoids network communication by running the focus driving process on the same computer. For the first configuration, we averaged 15 measurements to obtain 55 ms end to end latency, with a standard deviation of 5 ms. For the second configuration, with 9 measurements we observe an average of 31 ms latency with 2 ms standard deviation.

It could be possible to further reduce some of the sources of latency in our configurations. However, we believe many systems will have comparable latency and thus our tests provide a realistic indication of the challenges of automatic focus pulling with different kinds of motion capture. The relatively high end to end lag of the first configuration helps demonstrate the benefit of compensating for lag in scenarios where it might be difficult to reduce the lag (e.g., high-speed photography).

3.5 Summary

At this point, we have shown the full process of modifying a consumer level lens and controlling it through serial communication. We have also gone through the process of calibrating the focus and calculating the motor tick to focus conversion function. Lastly, we also found the other unknowns in the system such as the latency for different configurations.

In the following chapter, we use these measurements to design predictive filters to compensate for the measured latency. We will explore a wide variety of different models while taking intro consideration the limitations of our lens and the focus tracking system we presented in this chapter.

CHAPTER 4
Filter design and prediction

We use a Kalman filter to track the target state, from which we can predict its future position and the necessary focus distance. We estimate the noise from measurements, and the lag used for the prediction step is obtained as previously described in Section 3.4. We implement two simple process models for the filter: constant velocity, and constant acceleration. However, in the filter and during delay compensation we also accommodate some non-smooth phenomena such as collisions or constant deceleration from sliding friction until rest.

We track 3D motion capture points, and following a simple Kalman filter formulation, we use

$$x_k = Fx_{k-1} + Bu_k + w_{k-1} \tag{4.1}$$

$$z_k = Hx_{k-1} + v_{k-1}. \tag{4.2}$$

with a linear process F and measurement model H. We will typically not be able to know anything about how our target is being controlled, and therefore drop the Bu_k term. We assume that the process noise w has a zero mean normal probability distribution $N(0,Q)$ and we model the noise with one parameter σ (see below). Similarly, we use a single parameter λ to model the measurement noise v, which we assume to be a zero mean normal distribution $N(0,R)$ with covariance $R = \lambda^2 I$, where I is the 3-by-3 identity matrix. We discuss how we set the noise parameters later in this section.

4.1 Constant velocity model

The simplest process model we use is a constant velocity model. We maintain 3D position and velocity as the state of the target, $x = (p^T, v^T)^T$. Assuming the model and measurement noise are uncorrelated discrete white noise processes, we have

$$F = \begin{pmatrix} I & hI \\ 0 & I \end{pmatrix}, \; H = \begin{pmatrix} I & 0 \end{pmatrix}, \; Q = \sigma^2 \begin{pmatrix} \frac{1}{4}h^4 I & \frac{1}{2}h^3 I \\ \frac{1}{2}h^2 I & h^2 I \end{pmatrix} \tag{4.3}$$

where I is again the 3-by-3 identity matrix, and h is the sampling time step. This steady-state Kalman filter is also known as an α-β filter. This is the default model which works best in most non-specific situations. It can be used for actors moving around or for any unplanned irregular motion.

4.2 Constant acceleration model

For tracking focus on objects that are thrown, falling, or sliding to a stop, a constant acceleration model can be more appropriate. In this case, we include the acceleration in the state, $x = (p^T, v^T, a^T)^T$, and use model, measurement, and covariance matrices corresponding to an α-β-γ filter (see [Simon, 2006]).

$$F = \begin{pmatrix} I & hI & \frac{1}{2}h^2 I \\ 0 & I & hI \\ 0 & 0 & I \end{pmatrix}, \; H = \begin{pmatrix} I & 0 & 0 \end{pmatrix}. \tag{4.4}$$

4.3 Non-smooth process models

Tracking an object sliding to a stop on the floor requires a piecewise model because the object transitions between constant deceleration to zero acceleration at zero velocity. We implement this by testing for a sign change in the depth velocity of the tracked state. We prevent the filtered state estimate from overshooting by zeroing the velocity and acceleration, and resetting the estimate covariance to Q. We achieve this by modifying the filter state directly in our code when a certain

condition is hit. This ensures that the model correctly tracks the transition from sliding to sticking, but we also need to use the same non-smooth model in delay compensation. That is, the focus distance is an integral of the estimated state, forward in time, by the end-to-end latency. We compute the integral numerically while including a test at each step to check if the object has come to rest.

For the box pushing example in our results, note that the box is initially at rest. We let the Kalman filter estimate both the acceleration of the push and then the deceleration due to friction. As a result, we have defocus errors during the push. Alternatively, we could improve tracking with a simple constant velocity model when the box is held or pushed by a hand, and reinitialize the filter on release to immediately transition to deceleration due to frictional sliding.

A similar situation occurs during collisions. Our examples include a ballistic motion that ends with an inelastic collision. In this case, we use the position of the floor as a trigger to reset the covariance to Q and to zero the velocity and acceleration. Again, numerical integration starting from the estimated state and including an inelastic collision model provides delay compensation for the control of focus. For the start of the falling humanoid example in our results, we reset the covariance and set the acceleration to -9.8 m/s^2 at a given transition condition (the velocity exceeding a threshold).

4.4 Noise parameter optimization

The values for σ and λ have an important effect on the quality of the final results. We find the values experimentally by running an optimization over sample data, i.e., motion that was recorded in test runs, or *rehearsals*. Since the motion capture provides the position over the entire trajectory, we can know the position of the target in the future, as well as the positions predicted by our filter. In this manner, we compute the error in focus distance throughout each rehearsal. We then

search the space of σ and λ parameters for values which minimize this error using SciPy's `optimize` module (see Section 5.5).

4.5 Rigid object tracking

Camera motion requires a prediction of rigid motion to track the position and normal of the focal plane. Our motion capture system tracks both position and orientation. However, to simplify the problem of filtering orientations, we use only use the raw marker position measurements. We use independent Kalman filters and delay compensation in a first step, followed by a shape matching step to obtain the predicted frame [Müller et al., 2005]. We note that it would not be difficult to use an extended Kalman filter with linearized rotations to deal with camera rotation instead [Drews et al., 2013].

When tracking a camera, we set the origin to be on the focal plane, at a point approximately corresponding to the center of the lens. For an actor, in contrast, a point of interest within the frame can be selected, for instance, an eye.

4.6 Summary

We have now developed a model for our scene and various filters that are designed to work in different scenarios. These range from the basic constant velocity model, which is general enough to be useful in most cases, but we also looked at more complex non-linear filters for when the focus target follows a less arbitrary motion. We also looked at how to optimize the parameters in our filters, using the system measurements found earlier and recorded results.

We will next demonstrate how these filters perform using the setup described in Chapter 3 to get real-world measurements, as well as evaluate the filters synthetically with recorded motion capture data.

CHAPTER 5
Examples and Evaluation

We evaluate our predictive filters in two ways. First, we measure blur in videos recorded both with and without our predictive filter, as well as measuring blur in videos recorded with the Canon 70D follow focus mode. Second, given our measurement of the latency, we use motion capture trajectories to synthetically evaluate the performance of the filter without trying to measure the blur in video frames.

5.1 Standard viewing scenario

In this section, we use a standard viewing scenario to evaluate and compare our defocus results. All arcmin[1] blur measurements are defined with respect to the following setup.

Observers are assumed to be seated at a distance of 150 cm from a high definition 24-inch monitor. The stimulus on the display is 32.6 cm wide and 1200 pixels, or about 1.6 pixels per arcmin. This viewing angle and resolution defines the standard viewing scenario for this section.

5.2 Measuring blur in video

To get an objective idea of how well our filter is doing, we need to compare the blur in videos recorded with and without to filter and see if our solution actually reduces the blur in the recording. There are many blur metrics in the literature and a few were tested for this project. First, we tested a perceptual blur metric by Crete et al. [2007] which was validated with subjective tests and psychophysics functions. We have also experimented with a method presented by Pertuz et al. [2013] which

[1] An arcmin is a unit of angular measurement equal to $^1/_{60}$ degrees

explores metrics for focus. Lastly, we also look at a blur detection method presented by Shi et al. [2014] which is built on top of a huge blur perception dataset.

While these methods gave us great insight on blur metrics and blur perception, we found that we were able to achieve more precise and repeatable results by constructing a custom scene with a focus target and using that to compute the exact blur in each frame. We can see in Figure 5–1 an example of some of the blur metrics we experimented with being compared to our custom built blur measuring method. As we can observe, while the generic purpose blur metrics manage to get some of the patterns, they do not provide as much detail and consistency as our manual method.

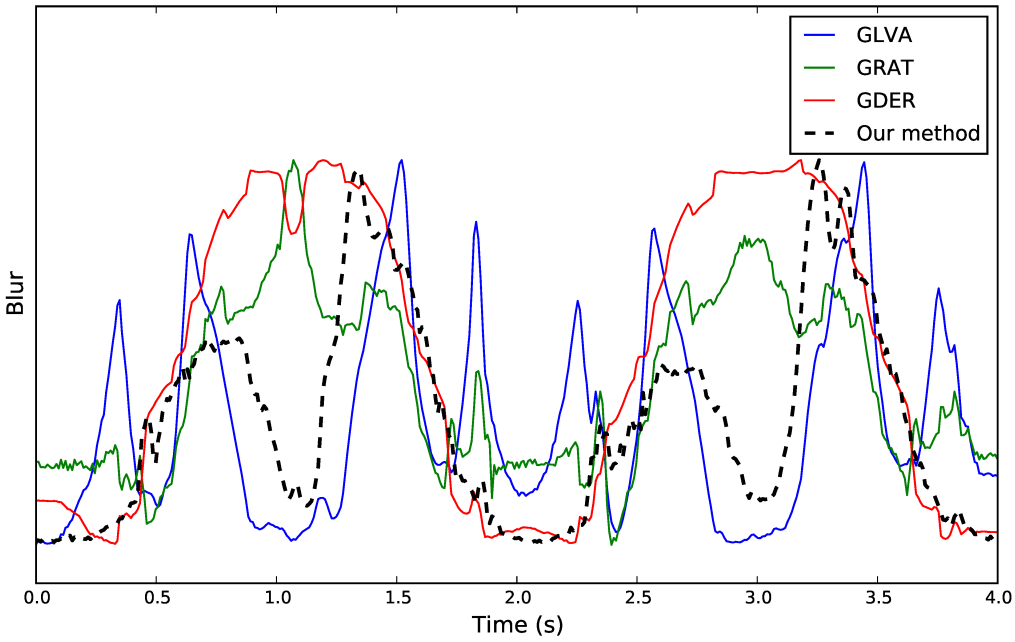


Figure 5–1: Comparison of the result of our custom blur measuring method versus general purpose blur metrics.

For this, we place a flat sheet with multiple sharp white to black edges in the center. We also place a small red and green squares on the target for tracking purposes, as seen in Figure 5–2). Our program automatically finds the green and red markers by segmenting based on color and averaging. We then crop an area of 100 horizontal lines between the markers. On each line we first identify the

black and white signal levels, and then search for the sub- pixel position of the 10% and 90% intensity levels. If noise in the image gives multiple crossings at either threshold, we average these crossings. We average the blur edge width on each line and multiply by 0.39 to estimate the standard deviation of an equivalent Gaussian blur [Smith, 1997]. Finally, we divide by the conversion factor of 1.6 pixels per arcmin for our standard viewing model defined in the psychophysics experiments to obtain values in arcmin, which allows for comparison with the results of our psychophysics experiments. The Matlab script used to perform this analysis can be found in Appendix B.

Figure 5–2: Our custom focus target with sharp white to black edges used to measure the blur in each frame of our focus test videos.

5.3 Computing blur from motion capture

After recording a video, the captured motion contains the full trajectory of the target. Therefore, knowing the system latency, we can evaluate how well our predictive filters perform.

While the difference between the filter prediction and measured depth is a useful measure of performance, we combine this with a camera depth of field model, and a viewing scenario in order to produce an estimate of the image blur in arcmin.

We use a 24-inch diagonal high definition display at a distance of 150 cm and a Canon 70D with an APS-C sensor that has a 1.6× crop factor. We use two different prime lenses, 28 mm and 85 mm, and we use both at f/1.8, which is the maximum aperture because this provides the most challenging focus scenario.

We use a simple thin lens model to approximate the circle of confusion for an out of focus target. Let S_1 be the distance to the focal plane in the scene, and let S_2 be the distance to the target. The diameter of the circle of confusion is given as

$$c = \frac{f}{N}\left|\frac{1}{S_1} - \frac{1}{S_2}\right|, \tag{5.1}$$

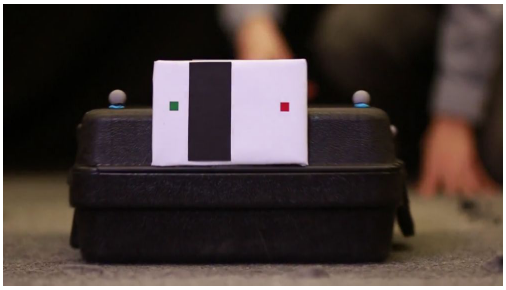
where the lens has f-number N and focal length f.

For the Canon 70D, to convert the circle diameter to high definition video pixels we multiply by $85.\bar{3}$, or 1920 pixels divided by 22.5 mm (which is the full frame width of 36 mm divided by the 1.6 crop factor). We subsequently convert the diameter of the circle of confusion to the standard deviation of its point spread function by multiplying by $1/\sqrt{2}$. Finally, we divide by 1.6 to convert from pixels to arcmin following our standard viewing model.

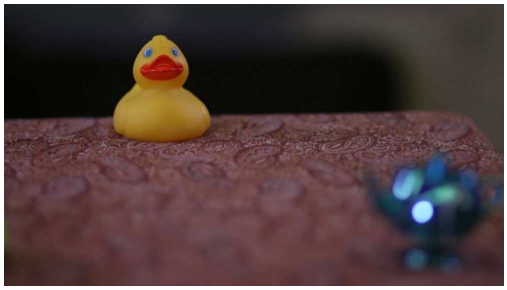
Note that this motion capture to arcmin conversion and our blur perception thresholds can help the assistant camera operator select an aperture that would allow blur caused by filter prediction errors to fall below the perceptual threshold, without the need to measure blur in raw video frames.

5.4 Examples

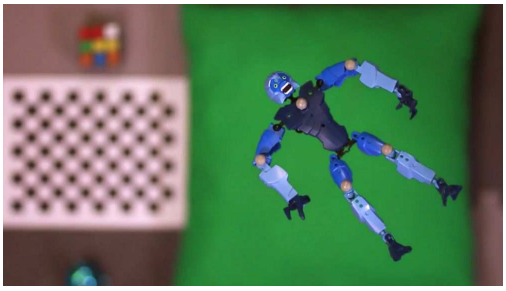
To test out our filters, we recorded many different examples, all of which can be seen in the provided video. A preview of the video can be seen in Figure 5–3. Some of these examples were used for measuring blur in videos, some were used for measuring blur synthetically using mocap data, while others simply there to demonstrate the limits of our system. We will now explain each example in more detail.



(a) Sliding box

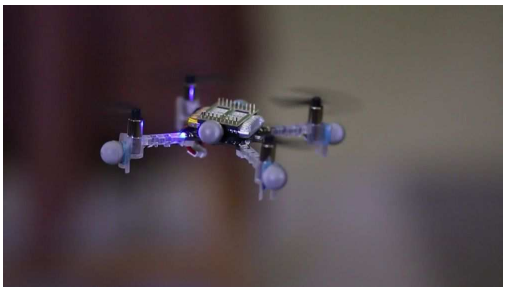


(b) Static objects

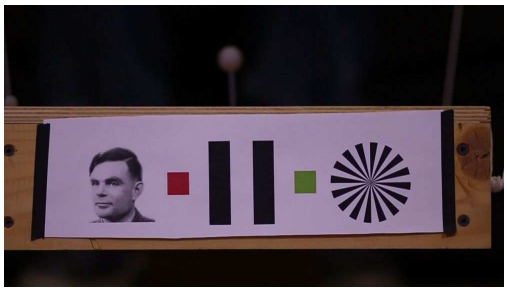


(c) Freefall

(d) Sudden motion



(e) Flying Quadrotor



(f) Pendulum motion

Figure 5–3: Example frames from recorded video using filtering and delay compensation for various scenarios. See supplementary video for the full demo of the filter in each scenario

5.4.1 Flying quadrotor

The first example we experiment with is that of a flying quadrotor, as seen in Figure 5–3(e). For this example, we use the simplest of our models, the constant velocity predictive filter. The arbitrary movement of a flying object or a moving human is hard to predict but slow enough for our base filter to do a satisfactory job. Trying to manually track focus for such a target is nearly impossible even for the most skillful focus pullers, especially when using a very shallow depth of field as we have in this example. Our tracking system, on the other hand, manages to keep the quadrotor in focus for the whole duration of the flight with no help whatsoever from the camera assistant.

5.4.2 Static objects

In this example, we set a couple objects on a table and this time, it is the camera holder who walks around the room. In this case, we use rigid body tracking to model the camera to account for rotational prediction. This again is a very basic scene, which normally would be fairly hard to properly focus on with a wide aperture, but as seen in the example video and in Figure 5–3(b), our system is able to switch between the different objects and keep the focus as the camera moves around in different directions.

5.4.3 Sudden motion

This example is to show how fast our system is able to track objects. We use the base constant velocity filter once again, but here, we have a person swing a red card very rapidly towards the camera, while our system tries keeping focus on it the whole way through. The card, as seen in Figure 5–3(d), stays in focus while the subjects' face quickly becomes blurred in the background. This example really shows the power of such a system and how easy it is to create powerful and impressive shots.

5.4.4 Pendulum motion

For this example, we set up a pendulum which oscillates with a period of 3.4 seconds and an amplitude of 0.7 meters. This is one of the examples we used to do our numerical blur evaluation as explained in the previous section. We attached our tracking marker on the front of the pendulum, as seen in Figure 5–3(f) and focus on it using the constant acceleration predictive filter, as it can very closely approximate the harmonic motion of the pendulum. We record the motion with different setups and then analyze the video for blur.

We plot a comparison of blur measurements in Figure 5–4. This compares the Canon 70D focus tracking to motion capture tracking, both with and without delay compensation to overcome 31 ms of latency. Each type was recorded and measured 3 times to check consistency. The built in focus, as expected, performs by far the worse, with peak blur around 10 arcmin. Using the mocap system with no prediction already gives a huge benefit, but we still see peak blurs that average 2 arcmin, which is still in the noticeable region. Once we use our predictive filters, as seen by the green line, the blur falls below the perceivable threshold of 1 arcmin. For this experiment, we used the constant acceleration model to best approximate the sinusoidal motion. Looking at the plot, we can confirm our intuition, which is the tracking system manages to catch up with the target at the two peaks of the oscillation, but struggles to keep up when the pendulum is moving the fastest. We also notice that the tracking struggles a bit more when the target is closest to the lens, as the focus is much more sensitive defocus in close range.

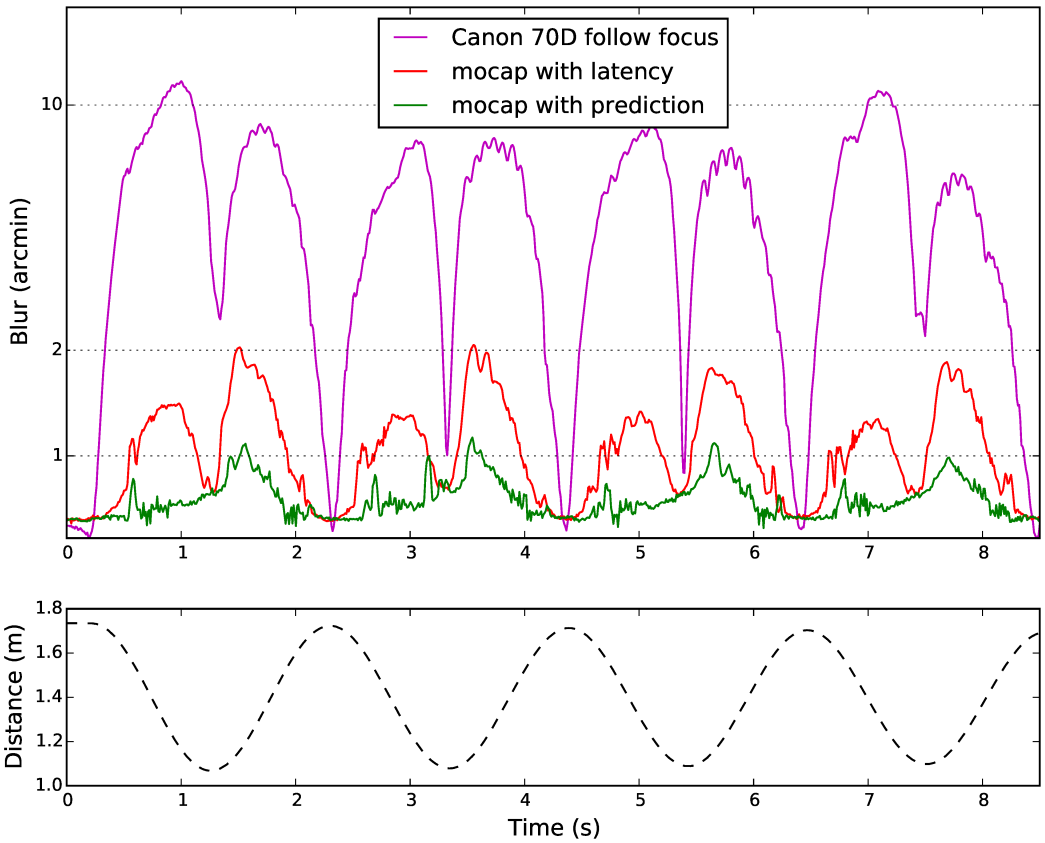


Figure 5–4: Blur in arcmin for our standard viewing model, measured in video of a pendulum recorded with the 85 mm lens, with a plot below showing the corresponding ground truth focus distance measured with motion capture. The purple curve shows measured blur for video recorded with the Canon 70D follow focus mode. The red curve shows the effect of latency on the blur. The green curve shows that predictive filtering gives good results.

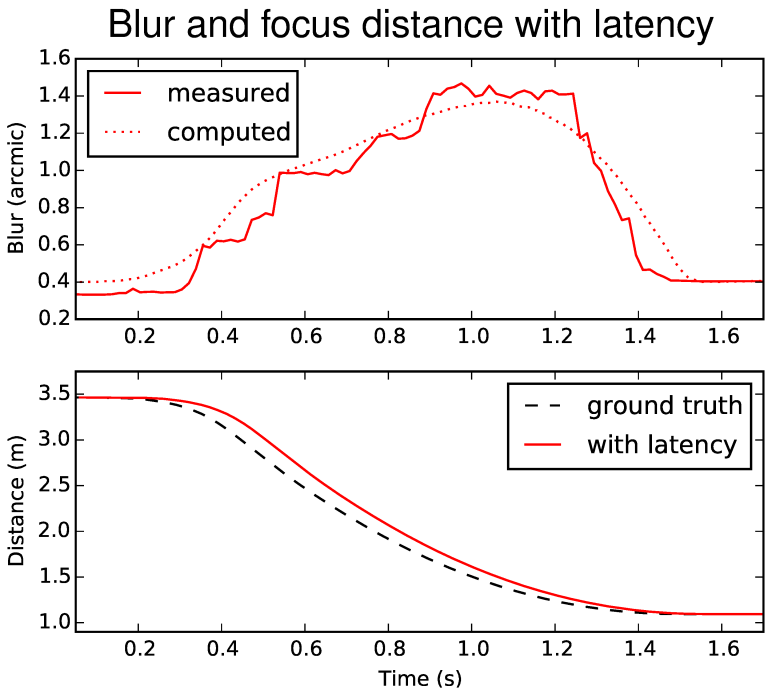
5.4.5 Sliding box

The next example we evaluate using our numerical measurement method is that of a sliding box. We record videos of a box sliding to rest on the ground, with the focus target attached on the front, as seen in Figure 5–3(a). For this experiment, we again use a constant acceleration model (decelerating due to friction), but we also add an extra non-linear constraint, which consists for limiting the velocity and acceleration to negative values. This prevents the prediction from overshooting when the box finally comes to a rest and avoids the extra defocus event which would usually happen at the end of the motion.

Figure 5–5 shows blur for motion capture driven autofocus with and without predictive filtering in our box sliding example. Notice how the green focus distance curve closely matches the motion capture ground truth, showing how well our filter can predict the motion of the sliding box. It is also good to see how the blur measured in the video approximately follows the same pattern as the blur predicted from the motion capture path and lens parameters. Lastly, looking at blur values, we observe that without filtering, we have a peak that is well above the 1 arcmin threshold for a long period of time. On the other hand, with the filtering, the blur line stays below 1 arcmin for the whole duration of the shot and therefore has no visible blur.

5.4.6 Freefall

Lastly, we have an example where the focus target is falling under the influence of gravity, then comes to an abrupt stop once it hits the ground. Sudden changes in motion like this are usually extremely difficult scenarios for a predictive system with latency, which is why we present this example. Here, for the falling section, we once again have a constant acceleration model, with a known acceleration (gravity, 9.8 m/s^2), but we add a non-linear event which is the ground collision. Since we



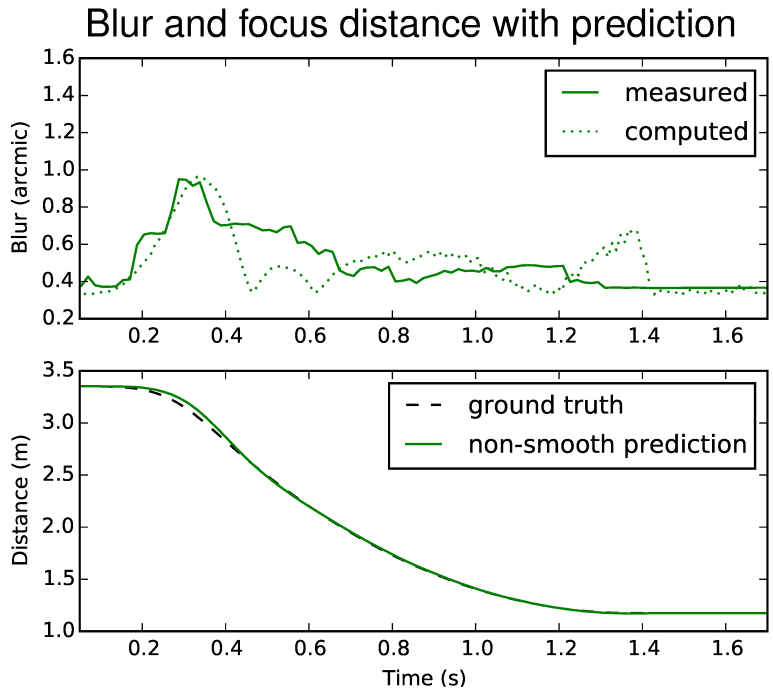


Figure 5–5: Plots show data for the sliding box recorded with the 85 mm lens and 33 ms latency. In each plot, the top part shows the blur measured and predicted, while the bottom part shows the ground truth position and the tracked position. The first plot shows the system with no filtering, whereas the second plot shows the system with our non-smooth predictive filter.

know where the ground plane is, we can actually predict when the target is going to hit the ground and better predict when to stop the focus motor.

In Figure 5–6, we plot the ground truth position of the target object, as well as the predicted motion without filter, with naive filter, and with our non-smooth filter. As expected, without a filter, the focus stays behind for the whole duration of the fall but does not overshoot at the end. On the other hand, the naive filter will track the fall properly, then overshoot after the collision. Meanwhile, as the synthetic simulation shows, the non-smooth filter is able to provide best of both worlds for this example and closely follow the ground truth.

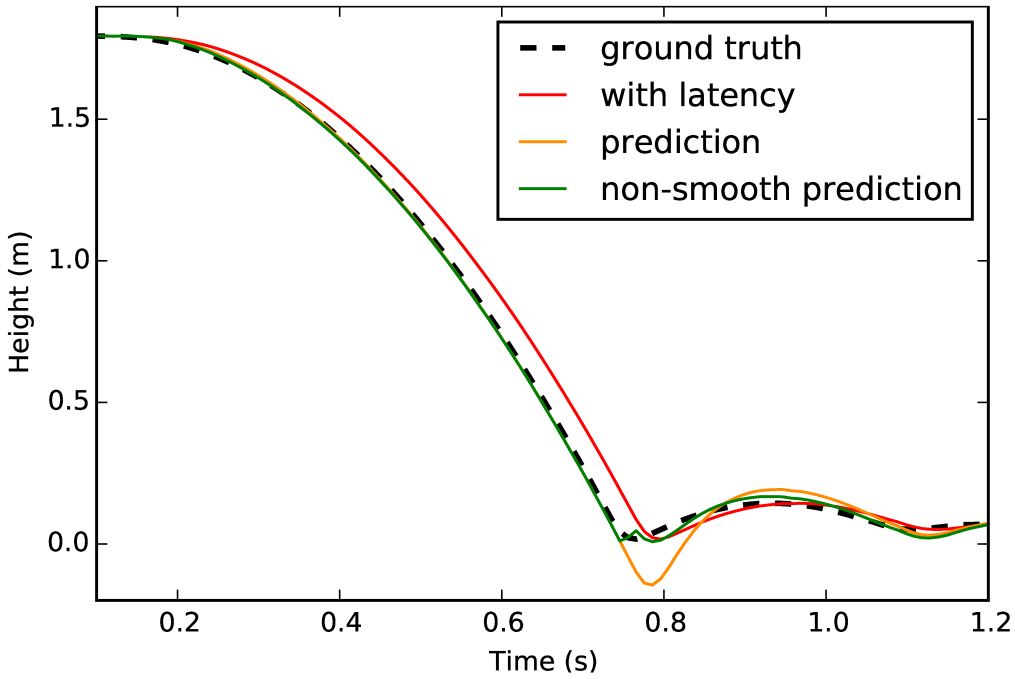


Figure 5–6: Comparison of height estimates in the example of tracking a falling object with inelastic collision.

5.5 Filter parameters

Figure 5–3 and the supplementary video show a number of different example scenarios. The optimal noise parameters of the filter depend on the latency and on how well the motion corresponds to a selected process model.

scenario	latency	model	norm	σ^2	λ^2
push + slide	55	acc.	L2	$10^{1.9}$	$10^{-8.9}$
push + slide	55	acc.	L1	$10^{1.4}$	$10^{-8.4}$
slide only	55	acc.	L1	$10^{1.4}$	$10^{-8.1}$
slide only	55	acc.	L1	$10^{1.4}$	$10^{-8.1}$
Crazyflie	31	vel.	L2	$10^{1.1}$	$10^{-8.1}$
Crazyflie	31	vel.	L1	$10^{2.6}$	$10^{-9.6}$
falling	55	acc.	L2	$10^{1.0}$	$10^{-6.0}$
falling	55	acc.	L1	$10^{1.6}$	$10^{-8.6}$

Table 5–1: Optimal noise parameters for different scenarios.

Note that non-smooth motion models play an important role in the final quality of many examples, and these model transitions are included within our optimization of noise parameters.

Figure 5–6 shows height estimates during the falling example. Without prediction, the red curve shows that latency leads to persistent defocus. The orange curve shows that prediction tracks well during the fall, but overshoots after the impact. The green curve shows that delay compensation with our non-smooth model matches closely the true height.

Table 5–1 gives an overview of the optimal parameters we find for a number of scenarios, including smooth motion (Crazyflie) and non-smooth motion (sliding to stop under friction, and falling with inelastic collision). Noise parameters correspond to standard SI units for the model and measurements (meters, seconds), and a motion capture rate of 100 Hz.

We explored both L1 and L2 minimization. While we speculated that optimization with the L1 norm might result in defocus errors that are harder to perceive, the optimal parameters suggest otherwise. Overall, the values are very similar across all of our tests. As such, we believe that the end-to-end latency is the most important factor influencing quality, then followed by the motion model.

5.6 Other use-cases

Alongside this work, there were a few other projects being worked on in our lab which ended up directly benefiting from the predictive filters developed here. Two such projects used a computer controlled quadrotor to perform artistic tasks, but each system had inherent delays causing inaccuracies. This is where our predictive Kalman filters came in to improve the results by compensating for the latency.

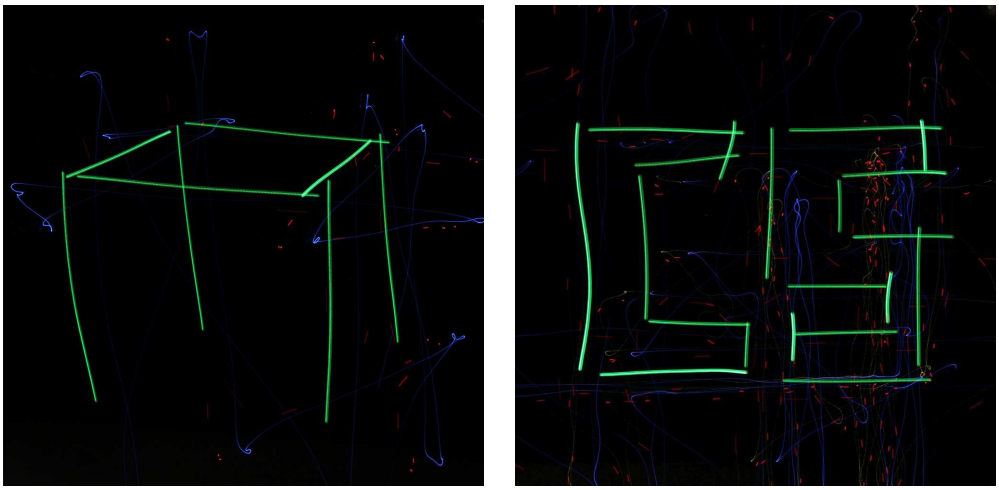
(a) LED attachment (b) Brush attachment

Figure 5–7: The quadrotor used for our projects with different attachments.

5.6.1 Light painting

The first project consisted of attaching an LED to a quadrotor and having it draw certain 3-dimensional shapes in the air while a stationary camera takes a long exposure photograph. An image of the quadrotor used with the LED attachment can be seen in Figure 5–7(a). Some preliminary results are shown in Figure 5–8. While these give a general idea of what such a system is capable of, we can already observe the errors caused by the end-to-end latency.

In this system, we use motion capture to accurately track the position of the quadrotor in order to know when to activate and deactivate the LED light. The issue arises due to the end-to-end delay from getting the position to sending a command to the LED, resulting in the lines being offset depending on the latency and the



(a) Table (b) CS (computer science)

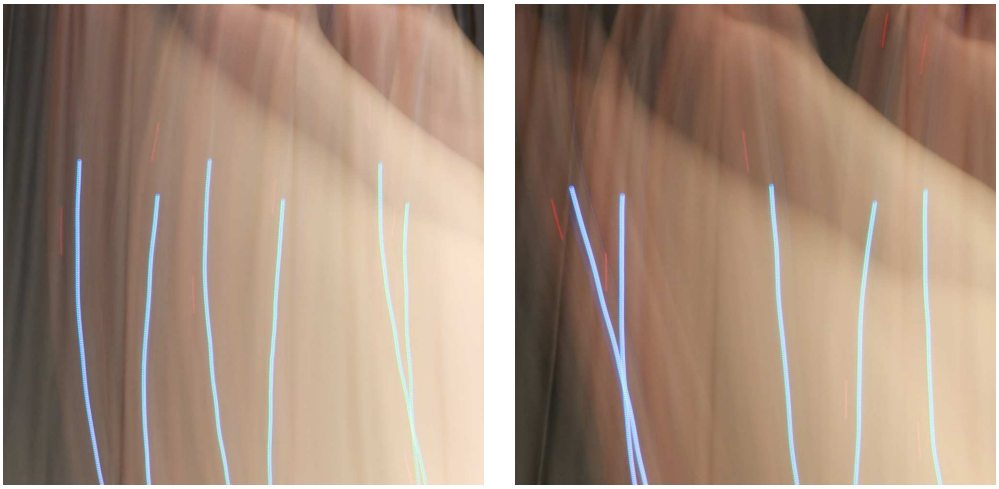
Figure 5–8: Long exposure photograph of a quadrotor drawing various shapes in the air with an attached LED. The shapes are roughly a meter in size.

velocity of our tracked quadrotor. This issue can be seen clearly in Figure 5–9, where the experimenter swings the quadrotor up and down, with the LED being programmed to turn on when below a certain height. As we can observe, without any filtering, the LED ends up turning on too late on the way down and turning off too late on the way up, resulting in a skewed boundary. On the other hand, once we apply the predictive filtering, we observe a clean straight boundary.

5.6.2 Stipple drawing

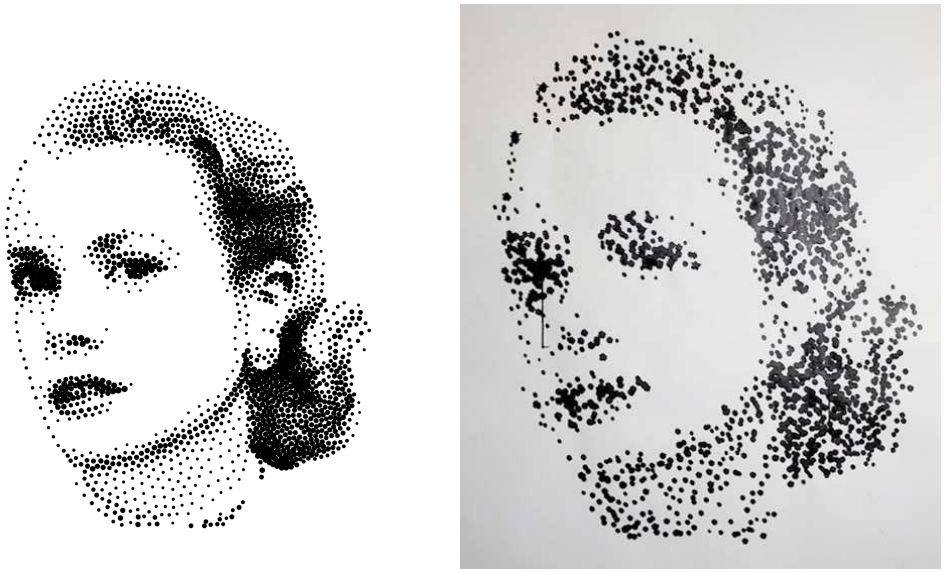
The next project consisted of attaching a brush to the quadrotor and having it create stipple drawings on a canvas by repeatedly colliding into it [Galea et al., 2016]. A picture of the modified quadrotor can be seen in Figure 5–7(b).

For this project, we first generate a stipple version of an input image and then compute an optimal order to draw the dots. The quadrotor, which is being tracked by the motion capture system, then draws the stipples one by one onto the canvas. An example generated and resulting stipple drawing can be seen in Figure 5–10. In this project, as the previous one, it is crucial to have accurate position information for the quadrotor in order to reduce the position error for each stipple.



(a) Without predictive filtering (b) With predictive filtering

Figure 5–9: Example of the error in light painting due to latency in the system and the same experiment performed with predictive Kalman filtering applied.



(a) Computer generated drawing (b) Quadrotor drawing

Figure 5–10: An example stipple drawing generated by our algorithm and the resulting drawing by the quadrotor.

5.7 Summary

We presented our setup and introduced the two methods used for evaluating the filters we have developed: measuring the blur in video footage of the filter in action and measuring expected blur from our filter running on motion capture data.

Next, we presented six different examples showcasing each of the filters we designed and in each case, we observed that using our predictive filters lowered the blur to below the perceivable threshold. We also observed that using models specific to the scene are sometimes necessary to stay below the threshold.

Lastly, we looked at two more use-cases for our predictive Kalman filters in other projects involving quadrotors. This shows how the findings in our work can have benefits in many other places. We will now conclude and present a few promising new directions for future works.

CHAPTER 6
Conclusion

We first present an automatic focus method that uses motion capture to control a modified electro-focus lens. We describe how to modify and calibrate a lens, how to communicate and control it using serial communication, as well as measure end-to-end latency in the complete system. We then present predictive filters for automatic focus control in various scenarios, based on the measurements in our system as well as psychophysics experiments. Next, we record a series of examples using the automatic focus system we built and evaluate those videos to numerically show how our predictive filters perform.

We found in our experiments that using a naive motion capture tracking system with no filtering while giving decent results, still had perceivable blur in more challenging scenarios. On the other hand, with our predictive filters, the peak blurs were reduced by half on average, putting them below detection thresholds according to psychophysics experiments, even with approximate motion models and moderate amounts of latency.

We looked at different models such as constant velocity, constant acceleration, and models with non-linear events. For each case, we showed how to design a custom filter and observed that there can be benefits to using scene specific models. In general, we found that the constant velocity model performs well enough in most situations.

From these results, it is clear that a system like this can have very beneficial use-cases in the movie industry and in helping with more complex and challenging shots. In the case of slow motion, every slight defocus event would get amplified even more, so it because much more crucial to have a precise and rapid focus

system. Such a task may be impossible to ever achieve by a human being and an automatic focus system like presented in this paper may be ultimately needed for the more extreme shots. It will allow for a whole new range of shots that were simply not possible before and give a new tool set for the director to explore.

6.1 Future work

We believe that it will worthwhile exploring the use of our predictive focus method with wireless magnetic motion capture sensors, which can be hidden on actors and objects, to investigate how the latency and noise of these sensors influence defocus of moving actors and objects. Our predictive filters would work great for addressing the added phase delay of noise filtering. It would also be interesting to look into the development of adaptive motion models and improved noise parameters for our filters. Having an automatic system which can choose and adapt which configuration to use depending on the scene would make this system far more user-friendly and simple to use.

As mentioned in the related works, the company Andra currently sells a similar system. While their setup ultimately lacks our filtering component, they instead have an interesting UI on a tablet providing the user with a better control of the setup and parameters. It would be interesting to explore such an interface in the future and see what can be done in that direction trying to automate as much as possible while still giving the director some control when needed.

Another subject worth looking as pointed out in an earlier chapter is working with more extreme lenses such as anamorphic ones. Currently, we assume that the target will always be in the center of the lens, and our model ignores distortions that happen in the outer region of the frame. For regular lenses, we found that the effects were negligible, but it would be interesting add that to our model to better approximate the focus distance in all regions of the frame.

Lastly, another direction we would like to explore is developing visually based metrics for evaluating blur in videos. We would need to run psychophysics trials on humans to evaluate the probability of seeing a blur of a certain length and size. Then, from that data, we would create a function that gives us the probability of a certain blur in a video to be perceived. Then, we could use this metric to run a more perceptually based optimization on our filters.

Appendix A

This is the Arduino script used to control the 28 mm and 85 mm Canon lenses. The timings were measured using an oscilloscope for our lenses and are needed to work properly. The lens uses a modified SPI protocol, meaning that we cannot use the built-in Arduino SPI. Instead, we re-implement our own in `send_signal`, which lets go of the clock after each message sent and waits for the lens to pull the current, signaling that it is done processing the command.

The clock speed for the Arduino also matters. For the 28mm lens, we originally used 82 KHz which worked fine, but for the 85mm lens, we require a faster 500 KHz clockspeed to communicate with the lens with the required timings.

```
void setup() {
  Serial.begin(115200);
  init_pins();
  init_lens();
}

void init_pins() {
  pinMode(LogicVDD_Pin, OUTPUT);
  pinMode(Lens2Cam_Pin, INPUT_PULLUP);
  pinMode(Cam2Lens_Pin, OUTPUT);
  pinMode(Clock_Pin, OUTPUT);
}

int starting_position = 0;
void init_lens() {
  digitalWrite(Clock_Pin, LOW);
  digitalWrite(LogicVDD_Pin, LOW);
  digitalWrite(Cam2Lens_Pin, LOW);
  delay(500);
  digitalWrite(Cam2Lens_Pin, HIGH);
  digitalWrite(LogicVDD_Pin, HIGH);
  digitalWrite(Clock_Pin, HIGH);
  delay(500);

  busypoll();

  starting_position = 0;
  starting_position = get_motor_position();
  Serial.println("ready");
}

void focus_nearest() {
  send_signal(0x06);
```

```
  send_signal(0x00);
}

void aperture_full() {
  send_signal(0x13);
  send_signal(0x80);
  send_signal(0x13); // second aperture_step cmd needed ...
  send_signal(0x80); // depending on current aperture pos
}

int get_motor_position() {
  send_signal(0xC0);
  byte hi = send_signal(0x00);
  byte lo = send_signal(0x00);
  signed short pos = ((hi & 0xFF) << 8) | (lo & 0xFF);
  return pos - starting_position;
}

void focus_step(int step) {
  if (step == 0) return;

  byte lb = step & 0xFF;
  byte hb = (step>>8) & 0xFF;

  send_signal(0x44);
  send_signal(hb);
  send_signal(lb);
  delayMicroseconds(500);
  busypoll();
}

// 85mm
float p0 =  1621.38579;
float p1 = -94715.8913;
float p2 = -3638133.32;
int F_MAX = 1680;

void focus_dist(float d) {
  float x = 1.0 / d;
  float y = p0 + x * (p1 + x * p2);
  int target = min(F_MAX, max(0, (int)y));

  int current = get_motor_position();
  while (current < -10 || current > 2000) {
    busypoll();
    current = get_motor_position();
  }

  focus_step(target - current);
}

void busypoll() {
  int count = 0;
  byte resp = 0x00;
  while (resp != 0xAA && count < 20) {
    send_signal_slow(0x0A);
    resp = send_signal_slow(0x00);
    count++;
  }
}
```

```
byte send_signal(byte sig) { // SPI command generator
  noInterrupts();
  byte rcv = 0;
  byte sig_ = sig;
  delayMicroseconds(100); // Delay observed empirically

  for (int i = 8; i > 0; i--) {
    // Falling edge
    CLR_DCLK; // digitalWrite(Clock_Pin, 0);
    if ((sig_ & 0x80) == 0x80) SET_DOUT; // digitalWrite(Cam2Lens_Pin, 1);
    else                       CLR_DOUT; // digitalWrite(Cam2Lens_Pin, 0);
    sig_ = sig_ << 1;
    SLEEP_400_NS; SLEEP_400_NS; NOP; // Pad LO duration to ~1us

    // Rising edge
    SET_DCLK;
    rcv = (rcv << 1) | POLL_DIN;
    SLEEP_400_NS; SLEEP_300_NS; NOP; // Pad HI duration to ~1us
  }

  SET_DOUT; // digitalWrite(Cam2Lens_Pin, HIGH);

  pinMode(Clock_Pin, INPUT_PULLUP);
  delayMicroseconds(15); // Empirically
  interrupts();

  while(!digitalRead(Clock_Pin));

  pinMode(Clock_Pin, OUTPUT);
  interrupts();
  return rcv;
}
```

Appendix B

This Matlab script was used to extract the blur from a video using our custom focus target and markers as presented in Chapter 5. The script first reads the frames from a video, uses the markers to crop the appropriate region. It finally extracts the blur amount from each frame by fitting the best blur kernel.

```
% Read video frames
vid = VideoReader('pendulum_filter.mp4');
frames = read(vid);
CR = [250.0, 250.0, 840.0, 400.0];
frames = frames(CR(2):CR(2)+CR(4),CR(1):CR(1)+CR(3),:,1:end);

gpos = zeros(size(frames, 4), 2);
rpos = zeros(size(frames, 4), 2);

% Get initial marker positions
imshow(frames(:,:,:,1));
h = impoint(gca, []);
rcenter = int32(wait(h));
h = impoint(gca, []);
gcenter = int32(wait(h));
mwidth = 50;
close;

% Track markers across all frames
for f=1:size(frames, 4)
    frame = frames(:,:,:,f);
    frame = double(frame)/255;

    gxs = max(gcenter(1) - mwidth, 1);
    gxe = min(gcenter(1) + mwidth, CR(3));
    gys = max(gcenter(2) - mwidth, 1);
    gye = min(gcenter(2) + mwidth, CR(4));

    rxs = max(rcenter(1) - mwidth, 1);
    rxe = min(rcenter(1) + mwidth, CR(3));
    rys = max(rcenter(2) - mwidth, 1);
    rye = min(rcenter(2) + mwidth, CR(4));

    R1 = frame(rys:rye,rxs:rxe,1);
    G1 = frame(rys:rye,rxs:rxe,2);
    B1 = frame(rys:rye,rxs:rxe,3);

    R2 = frame(gys:gye,gxs:gxe,1);
    G2 = frame(gys:gye,gxs:gxe,2);
    B2 = frame(gys:gye,gxs:gxe,3);

    rmarker = medfilt2(R1 - 0.5 * (G1 + B1) > 0.24, [5 5]);
    gmarker = medfilt2(G2 - 0.5 * (R2 + B2) > 0.04, [5 5]);

    [R,C] = find(rmarker);
    rpos(f,:) = mean([C,R]) + double([rxs,rys]);
    [R,C] = find(gmarker);
    gpos(f,:) = mean([C,R]) + double([gxs,gys]);

    gcenter = int32(gpos(f,:));
    rcenter = int32(rpos(f,:));
    mwidth = (gcenter(1) - rcenter(1)) / 5;
end

gpos = [smooth(gpos(:,1), 20), smooth(gpos(:,2), 20)];
rpos = [smooth(rpos(:,1), 20), smooth(rpos(:,2), 20)];
```

```
dist = gpos - rpos;
dist = sqrt(dist(:,1).^2 + dist(:,2).^2);
min_dist = min(dist);
mid = size(frames(:,:,:,1))/2;
mid(3) = 0;

sigmas = zeros(1, size(frames, 4));

% Crop each frame given the marker positions
for f=1:size(frames, 4)
    rd = round(rpos(f,:));
    gd = round(gpos(f,:));
    frame = frames(:,:,:,f);

    dif = gd - rd;
    angle = atan2(dif(2), dif(1));
    if angle < -pi/2
        angle = angle + pi;
    elseif angle > pi/2
        angle = angle - pi;
    end

    frame = imrotate(frame, rad2deg(angle)/2, 'bilinear');

    midx = floor((rd(1) + gd(1))/2);
    dist = floor(2 * (gd(1) - rd(1)) / 5);
    minx = midx - dist;
    maxx = midx + dist;
    miny = rd(2) - 50;
    maxy = rd(2) + 50;

    frame = frame(miny:maxy, minx:maxx, :);
    frame = rgb2gray(frame);

    curve = sum(frame);
    curve = (curve - min(curve)) / (max(curve) - min(curve));
    cx = find(diff(sign(curve-0.50001)));
    assert(size(cx, 2) == 4);

    h = size(frame, 1);
    sigma = zeros(1, 4 * h);
    for i=1:size(frame,1)
        curve = double(frame(i,:));
        for j=1:4
            white = mean(curve(cx(j)-40:cx(j)-30));
            black = mean(curve(cx(j)+30:cx(j)+40));
            subcurve = curve(cx(j)-30:cx(j)+30);

            if white < black
                tmp = white;
                white = black;
                black = tmp;
            end
            subcurve = (subcurve - black) / (white - black);

            if subcurve(1) < subcurve(end)
                subcurve = 1 - subcurve;
            end
            sigma(i + (j - 1) * h) = rise_distance(subcurve);
        end
    end

    sigmas(f) = mean(sigma);
    sigmas(f)
end
```

References

Andrew Adams, Eino-Ville Talvala, Sung Hee Park, David E. Jacobs, Boris Ajdin, Natasha Gelfand, Jennifer Dolson, Daniel Vaquero, Jongmin Baek, Marius Tico, Hendrik P. A. Lensch, Wojciech Matusik, Kari Pulli, Mark Horowitz, and Marc Levoy. The frankencamera: An experimental platform for computational photography. In *ACM SIGGRAPH 2010 Papers*, SIGGRAPH '10, pages 29:1–29:12, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0210-4. doi: 10.1145/1833349.1778766. URL http://doi.acm.org/10.1145/1833349.1778766.

Ronald Azuma and Gary Bishop. Improving static and dynamic registration in an optical see-through HMD. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 197–204. ACM, 1994.

J. Baldwin, A. Basu, and Hong Zhang. Predictive windows for delay compensation in telepresence applications. In *IEEE International Conference on Robotics and Automation*, volume 4, pages 2884–2889, May 1998. doi: 10.1109/ROBOT.1998.680629.

Yosuke Bando, Henry Holtzman, and Ramesh Raskar. Near-invariant blur for depth and 2D motion via time-varying light field analysis. *ACM Trans. Graph.*, 32(2): 13:1–13:15, April 2013. ISSN 0730-0301. doi: 10.1145/2451236.2451239. URL http://doi.acm.org/10.1145/2451236.2451239.

Laurent Belcour, Cyril Soler, Kartic Subr, Nicolas Holzschuch, and Fredo Durand. 5D covariance tracing for efficient defocus and motion blur. *ACM Trans. Graph.*, 32(3):31:1–31:18, July 2013. ISSN 0730-0301. doi: 10.1145/2487228.2487239. URL http://doi.acm.org/10.1145/2487228.2487239.

David C. Burr and Michael J. Morgan. Motion deblurring in human vision. *Proceedings of the Royal Society B: Biological Sciences*, 264(1380):431–436, 1997.

Addison Chan, Rynson W. H. Lau, and Beatrice Ng. Motion prediction for caching and prefetching in mouse-driven DVE navigation. *ACM Trans. Internet Technol.*, 5(1):70–91, February 2005. ISSN 1533-5399. doi: 10.1145/1052934.1052937. URL http://doi.acm.org/10.1145/1052934.1052937.

Frederique Crete, Thierry Dolmiere, Patricia Ladret, and Marina Nicolas. The blur effect: perception and estimation with a new no-reference perceptual blur metric. *Proc. SPIE*, 6492:64920I–64920I–11, 2007. doi: 10.1117/12.702790. URL http://dx.doi.org/10.1117/12.702790.

J. Demers. *GPU Gems*, chapter Depth of field: A survey of techniques. Addison-Wesley, 2004.

T. M. Drews, P. G. Kry, J. R. Forbes, and C. Verbrugge. Sequential pose estimation using linearized rotation matrices. In *Computer and Robot Vision (CRV), 2013 International Conference on*, pages 113–120, May 2013. doi: 10.1109/CRV.2013.33.

Brendan Galea, Ehsan Kia, Nicholas Aird, and Paul G Kry. Stippling with aerial robots. In *Proceedings of the Joint Symposium on Computational Aesthetics and Sketch Based Interfaces and Modeling and Non-Photorealistic Animation and Rendering*, pages 125–134. Eurographics Association, 2016.

A.V.P. Grosset, M. Schott, G.-P. Bonneau, and C.D. Hansen. Evaluation of depth of field for depth perception in DVR. In *Visualization Symposium (PacificVis), 2013 IEEE Pacific*, pages 81–88, Feb 2013. doi: 10.1109/PacificVis.2013.6596131.

Robert T. Held, Emily A. Cooper, James F. O'Brien, and Martin S. Banks. Using blur to affect perceived distance and size. *ACM Trans. Graph.*, 29(2):19:1–19:16, April 2010. ISSN 0730-0301. doi: 10.1145/1731047.1731057. URL http://doi.acm.org/10.1145/1731047.1731057.

Wenzel Jakob, Jonathan T. Moon, and Steve Marschner. Capturing hair assemblies fiber by fiber. *ACM Trans. Graph.*, 28(5):164:1–164:9, December 2009. ISSN 0730-0301. doi: 10.1145/1618452.1618510. URL http://doi.acm.org/10.1145/1618452.1618510.

S.J. Julier and J.K. Uhlmann. Fusion of time delayed measurements with uncertain time delays. In *American Control Conference, 2005. Proceedings of the 2005*, pages 4028–4033 vol. 6, June 2005. doi: 10.1109/ACC.2005.1470607.

Andrew Kiruluta, Moshe Eizenman, and Subbarayan Pasupathy. Predictive head movement tracking using a kalman filter. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 27(2):326–331, 1997.

Sungkil Lee, Elmar Eisemann, and Hans-Peter Seidel. Real-time lens blur effects and focus control. *ACM Trans. Graph.*, 29(4):65:1–65:7, July 2010. ISSN 0730-0301. doi: 10.1145/1778765.1778802. URL http://doi.acm.org/10.1145/1778765.1778802.

Dikai Liu, Lingfeng Wang, and Kay Chen Tan. *Design and control of intelligent robotic systems*, volume 177. Springer, 2009.

G. Mather and D. R. R. Smith. Blur discrimination and its relation to blur-mediated depth perception. *Perception*, 31(10):1211–1219, 2002.

CH Messom, G Sen Gupta, S Demidenko, and Lim Yuen Siong. Improving predictive control of a mobile robot: Application of image processing and kalman filtering. In *Instrumentation and Measurement Technology Conference, 2003. IMTC'03. Proceedings of the 20th IEEE*, volume 2, pages 1492–1496. IEEE, 2003.

Matthias Müller, Bruno Heidelberger, Matthias Teschner, and Markus Gross. Meshless deformations based on shape matching. *ACM Transactions on Graphics (TOG)*, 24(3):471–478, 2005.

S Nayar. Computational camera: Approaches, benefits and limits. *Rapport technique. Cité*, page 6, 2011.

Ren Ng. Fourier slice photography. *ACM Trans. Graph.*, 24(3):735–744, July 2005. ISSN 0730-0301. doi: 10.1145/1073204.1073256. URL http://doi.acm.org/10.1145/1073204.1073256.

U. Orguner and F. Gustafsson. Target tracking using delayed measurements with implicit constraints. In *Information Fusion, 2008 11th International Conference on*, pages 1–8, June 2008.

Phillip T. Pasqual and Jacob O. Wobbrock. Mouse pointing endpoint prediction using kinematic template matching. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, pages 743–752, 2014. ISBN 978-1-4503-2473-1. doi: 10.1145/2556288.2557406. URL http://doi.acm.org/10.1145/2556288.2557406.

Said Pertuz, Domenec Puig, and Miguel Angel Garcia. Analysis of focus measure operators for shape-from-focus. *Pattern Recognition*, 46(5):1415 – 1432, 2013. ISSN 0031-3203. doi: http://dx.doi.org/10.1016/j.patcog.2012.11.011. URL http://www.sciencedirect.com/science/article/pii/S0031320312004736.

J. Shi, L. Xu, and J. Jia. Discriminative blur detection features. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2965–2972, June 2014. doi: 10.1109/CVPR.2014.379.

Dan Simon. *Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches*. Wiley-Interscience, 2006. ISBN 0471708585.

Steven W. Smith. *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Publishing, San Diego, CA, USA, 1997. ISBN 0-9660176-3-3.

Murali Subbarao and Gopal Surya. Depth from defocus: A spatial domain approach. *International Journal of Computer Vision*, 13(3):271–294, 1994. ISSN 0920-5691. doi: 10.1007/BF02028349. URL http://dx.doi.org/10.1007/BF02028349.

Kartik Venkataraman, Dan Lelescu, Jacques Duparré, Andrew McMahon, Gabriel Molina, Priyam Chatterjee, Robert Mullis, and Shree Nayar. PiCam: An ultra-thin high performance monolithic camera array. *ACM Trans. Graph.*, 32(6): 166:1–166:13, November 2013. ISSN 0730-0301. doi: 10.1145/2508363.2508390. URL http://doi.acm.org/10.1145/2508363.2508390.

B. Wang and K. J. Ciuffreda. Blur discrimination of the human eye in the near retinal periphery. *Optometry and Vision Science*, 82:52–58, 2005.

Andrew B. Watson and Albert J. Ahumada. Blur clarified: A review and synthesis of blur discrimination. *Journal of Vision*, 11(5):10, 2011. doi: 10.1167/11.5.10. URL +http://dx.doi.org/10.1167/11.5.10.

Jean Wlodarski. Canon EF-S protocol and electronic follow focus. https://pickandplace.wordpress.com/2011/10/05/canon-ef-s-protocol-and-electronic-follow-focus/, 2011.

Changyin Zhou and Shree K Nayar. Computational cameras: Convergence of optics and processing. *IEEE Transactions on Image Processing*, 20(12):3322–3340, 2011.