Juggling the Effects of Latency: Motion Prediction Approaches to Reducing Latency in Dynamic Projector-Camera Systems

Jarrod Knibbe [1,2] **Hrvoje Benko** [1] **Andrew D. Wilson** [1]

[1] Microsoft Research, USA
{benko, awilson}@microsoft.com

[2] Department of Computer Science
University of Bristol, UK
jarrod.knibbe@bristol.ac.uk

Figure 1. The Juggling Display is a custom projector-camera system demonstrating our motion prediction strategies for reducing the effects of latency on projection alignment. Through prediction, our system improves target illumination by 30%.

ABSTRACT

Projector-camera (pro-cam) systems afford a wide range of interactive possibilities, combining both natural and mixed-reality 3D interaction. However, the latency inherent within these systems can cause the projection to 'slip' from its intended target, detracting from the overall experience. Because of this, pro-cam systems have typically shied away from truly dynamic scenarios. In turn, research has been exploring latency reduction techniques across a range of domains, but these techniques typically focus on custom hardware, limiting their widespread adoption. We explore software-only predictive approaches to minimize the effects of latency in pro-cam systems. In this paper, we focus our predictive approaches on real-world objects under fast motion and on-body projection, improving projection accuracy on fast moving targets. Alongside this we explore automatic latency measurement techniques, allowing our system to determine and account for its own latency. We detail predictive approaches and provide results of a series of empirical investigations; achieving a 37% improvement in projection accuracy on objects in free flight (at speeds approaching 5m/s), and a 43% improvement in on body projection (with movement circa 1.5m/s). Through our work we aim to facilitate the wider exploration of pro-cam systems for 3D interaction in dynamic settings and showcase the accuracy achievable with off-the-shelf hardware.

Author Keywords: Latency; projection lag; projector-camera system, mixed-reality interaction, natural 3D interfaces.

Index terms: H.5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces

1. INTRODUCTION

Projector-camera (pro-cam) systems afford a wide range of interactive possibilities, including mixed-reality games (e.g. [1], [2]), interaction-anywhere (e.g. [3], [4]) and motion tutorial systems (e.g. [5]–[7]). All of these interactive systems are subject to the effects of latency, whether in visual delays when interacting with virtual objects (e.g. [2]) or projection misalignment when overlaying graphics on moving physical objects (e.g. [8], [5]). These misalignments and delays all result in the projection 'slipping' from its expected position and can easily have an adverse impact on the immersive experience. In order to avoid this projection 'slip', the speed of motion in pro-cam systems is typically heavily constrained and truly active scenarios have been avoided. For example, on person projection for coaching has been restricted to static pose guidance [6] and slow-motion tasks [5].

Pro-cam system latency is a combination of the latencies of each individual component, including: shutter delay, on-camera image processing, data transfer, tracking, projector buffering etc. Previous work has been conducted to reduce system latency through customized hardware (e.g. [9]) or advanced multi-camera tracking systems (e.g. [10]), but the requirement for significant expertise renders this approach at odds with the lightweight, easily-adoptable development approaches currently favored by both the enthusiast and research communities (as supported by readily available depth cameras such as the Microsoft Kinect). As a result of this, we explore the feasibility of software-only prediction approaches to

combatting the effects of latency. Through our work we aim to provide methods for improving projection alignment in pro-cam systems using off-the-shelf hardware, in turn encouraging further exploration of dynamic pro-cam systems and facilitating a wider range of interactive, mixed-reality experiences.

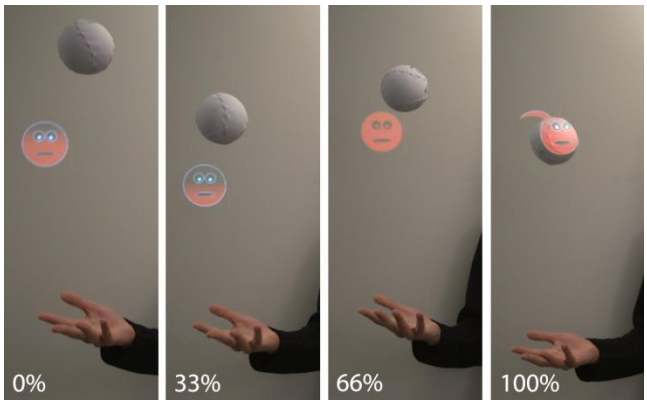

Figure 2. Images illustrating the visible effect of counteracting latency in our system. From the left, each subsequent image shows the result of a reduction in pro-cam latency of an additional 33%.

In this paper we focus on using motion prediction to reduce pro-cam latency, enabling accurate projection on fast-moving physical objects. Within this, we consider two example scenarios: objects in free flight and on-body projection (exploring prediction methods for human motion). We also use these domains to better situate our approaches and provide an opportunity for evaluation.

We explore different levels of motion predictability and provide methods that ensure promising projection alignment results with no requirement for hardware changes. We present a lightweight latency measurement process (alongside measurement values to act as a guideline for current hardware configurations) and detail a system that automatically measures and adapts to its own latency. Through two empirical evaluations we show an improvement in projection accuracy on a fast moving target from 14% to 50% of individual ball's flight time and a 43% increase in on-body projection accuracy.

2. RELATED WORK

Understanding and combatting latency in interactive systems is a popular area of work. We highlight the effects of latency on existing pro-cam research and draw upon research on hardware and software based latency reduction.

2.1. Pro-Cam Systems Affected by Latency

Within the scope of our work, we review pro-cam systems that demonstrate the effects of latency on interaction, whether acknowledged or not, such as in OmniTouch [3], LightGuide [5], YouMove [6], and MirageTable [11]. For example, in LightGuide [5], an instructive system to help guide users through hand motion tasks via on-body projections, participants are limited to movements of 30mm/s in order to maintain projection alignment. It becomes quickly apparent that this is unnaturally slow for the completion of most tasks. Building on LightGuide, YouMove provides a whole-body motion training system [6]. However, where LightGuide constrained users' movement speeds, YouMove delivers motion training through a pose-by-pose approach. While not specifically addressed, by avoiding real-time motion training and opting for pose-by-pose, the effects of latency on performance feedback could be significantly reduced. In a different domain, the OmniTouch [3] video shows the effects of latency on projection alignment when overlaying a number-pad on a piece of paper and when tracking the user's fingers across their hand. While these latency effects do not preclude the use of projection mapping, they serve to constrain the user's performance.

2.2. Hardware-based Latency Reduction

The effects of latency and frame-rate are important topics across a range of domains. For example, latencies in head tracking have very negative effects on the experience of Augmented and Virtual Reality (e.g., [12]–[14]). Papadakis et al. [15] minimize latency in head-tracked immersive simulations by reducing buffering latency in their display hardware, achieving a reduction in overall system latency of 50%.

In Lumospheres [10], Yamaguchi et al. present a hardware optimization approach to accurately project on balls under projectile motion. Our work is complementary to this and we build upon it in several ways. Firstly, Yamaguchi et al use 6 synchronized cameras capturing at 250Hz. We present a software-only approach that utilizes a single off-the-shelf depth camera capturing at 30Hz. We explore different levels of predictability, presenting a range of solutions, with examples across 2 different scenarios. Through this, we present a solution that applies broadly across a range of interactive domains and thus supports the wider transition of pro-cam systems to dynamic settings. Finally, by examining a similar scenario with a different focus (a *Juggling Display*), we can highlight the cost-accuracy tradeoffs that play a key role in this domain.

Similarly to the hardware approach of LumoSpheres, Okumura et al. developed a low latency camera for ball tracking [16]. This involves an intricate series of 'saccade mirrors' [17] and a camera capable of capture and processing at >1000Hz in order to maintain a ball position in the center of the frame. In a different domain, Ng et al., use novel hardware optimization to control the latency of touch screen devices down to approximately 1ms [9]. While users were able to perceive additional latency improvements below 10ms, further reduction below this point had minimal impact on task performance [18].

2.3. Software-based Latency Reduction

Xia et al. seek to find a camera and software-based approach for latency reduction on touchscreens [19]. They use a high-speed (120Hz) tracking camera and finger markers to track user finger movement. While their addition of a camera and finger markers would suggest a hardware-based approach, it is their in-software methods that are most relevant to our work, thus we include this as a software-based approach. Based on collected training data, Xia et al. estimate touch down locations and trigger device interactions in advance. Our work builds on Xia et al.'s principles in 2 key ways to explore latency reduction in pro-cam systems. Firstly, as Xia et al.'s work focused on a touchscreen with a known interface, prior knowledge of possible target locations could increase the accuracy of their prediction. We explore prediction within a less constrained environment, where the scope of motion is much greater (whole body movement) and no prior knowledge of target locations is available. Secondly, instead of applying an average user model for prediction, we learn from each user's individual approach, drawing on per-user expertise to provide a more personalized prediction.

A number of attempts have been made to explore and predict projectile motion. Kitani et al. [20] place a camera inside a ball and

use image processing to determine its speed of rotation, triggering the camera at precise moments to capture the scene below. There is a large body of work on the prediction of projectile motion within a military setting. For example, in [21], Fairfax et al, combine low-cost sensors and cameras into an Extended Kalman filter to predict object landing zones.

Our approach of using the Kalman Filter for prediction of motion in the future is similar to Liang et al. [13], who compensate for the delay in orientation data when head-tracking, as well as Friedman et al. [22], who predict collisions between drumsticks and virtual drums, to reduce the sound latency.

3. COMBATTING PRO-CAM LATENCY WITH MOTION PREDICTION

To reduce the effects of latency on projection misalignment in dynamic scenarios we focus on using motion prediction to model and derive the future states of objects. This enables us to model where an object will be and project on its future location, taking into account any system latency. Before outlining our example dynamic scenarios and predictable motion categories, we examine example pro-cam latency and clarify additional sources of projection slip.

Figure 3. The Juggling Display pro-cam unit consisting of InFocus IN1503 projector and a Kinect for Windows camera.

3.1. Estimating Pro-Cam Latency

To gain an understanding of end-to-end latency in pro-cam systems, we measured the latencies of 9 projectors (Dell 4320, Infocus IN1503, BenQ 720, LG HX350T, BenQ W1080ST, Infocus LP70, NEC VT46, Infocus IN1102 – all projecting at 60Hz) when paired with Microsoft's Kinect for Windows (30Hz).

In the spirit of our non-hardware augmented approach, we adopted an easy to implement frame-counting technique (as opposed to the more complex, hardware augmented, sub-frame accuracy achieved by Steed [23] and others [12], [24]). We capture a tennis ball in free fall with the Kinect and re-project the captured image back onto a co-planar surface. Using an additional high speed camera (120Hz), we capture both the real and projected tennis ball simultaneously and calculate the differences in position (given known refresh rates) to gain a ballpark latency measurement.

Over all of our projectors, the average latency with the Kinect camera (when processing color and depth) was 102.5ms (std. dev. 6ms). By processing only the color image, this latency reduced on average by 10%. While not directly relevant to our work (as we utilize both the color and depth images), this reduction highlights the importance of careful design and implementation decisions when developing systems of this style.

3.2. Sources of Projection Slip

In this work we explore latency as the principle cause of projection slip in systems involving dynamic motion. However, it is important to acknowledge other factors that contribute to projection misalignment.

Throughout our work, we utilise a first generation Kinect for Windows camera as it enables easy 3D registration of our scene and is popular in work of this kind (e.g. [5], [6]). The Kinect itself is subject to a range of errors. First, the color and IR cameras may be subject to inadequate calibration, resulting in inaccurate conversion between world- and camera-space [25]. Second, the depth measurements degrade increasingly with the square of the depth [25]. At a depth around 2m, Kinect is reportedly accurate +/- 1cm [26] (though this improves if averaged over time and can be further improved through morphological filtering [27]). Thirdly, both of the cameras utilise an electronic 'rolling' shutter which builds the image from the top down, resulting in the elongation of an object' representation when under motion. The extent of this elongation is relative to the object's motion. In our juggling scenario, the elongation of the ball's image changes significantly during flight as our ball's velocity decreases towards the zenith before increasing again towards the catch, thus introducing further measurement (tracking) error. Finally, the color and depth images are not time-synchronized, introducing further error when considering them side-by-side.

Alongside camera errors, there exist errors across the pro-cam system as a whole. First, there is an error as a result of the unpredictable interaction between the refresh rates of our camera and our projector which do not run on a synchronized clock. For example, if the image capture rate is not perfectly aligned to the projector refresh rate, it is possible that the result will be buffered and wait for one extra projector frame (16ms) before being displayed. While the effects of this synchronization could be reduced through the use of additional hardware technology such as NVidia's GSYNC, the requirement for additional hardware renders it outside the scope of our software-only approach. Finally, while a careful calibration procedure between the camera and projector is conducted, there also exist errors here.

4. PROJECTION ON FAST MOVING PHYSICAL OBJECTS

In order to focus our work on enabling dynamic pro-cam systems through software-based latency reduction, we explore two example domains. We present a range of general approaches that can be used to predict motion and provide practical examples in these domains. Through prediction we seek to minimize the effects of system latency and maximize on-target projection time.

4.1. *Scenario 1:* Objects in Free Flight; the *Juggling Display*

We develop a *Juggling Display*, a prototype pro-cam system where juggling balls are projected on, augmenting the juggler's performance with additional graphics (Figure 3).

In a typical pro-cam system, a 30Hz camera captures the scene, a computer tracks and renders graphics, and a 60Hz projector displays back onto the scene. As our preliminary investigation has shown, latency here is typically in the region of 100ms. Now imagine a juggler performing standard 3-ball juggling (as in Figure 1). The juggling balls are small and fast moving, with launch speeds easily exceeding 5m/s, resulting in 50cm of projection slip at launch. At the zenith of the ball's trajectory fleeting alignment occurs due to reduction in velocity, but this is short-lived as the ball quickly begins to accelerate downwards and the projection slip again increases. Without any motion prediction, only a small portion of the ball's flight is illuminated (14% - as shown in our

results). We explore latency reduction approaches to maximizing the possible display time during the ball's flight. Juggling provides a good target scenario for our exploration, as it includes both fast motion and a range of predictable features (including the ball's flight path and the juggler's hand motion – as we explain later). In this example, we use the Juggling Display simply as a visually compelling scenario, but it could also be used as a method for adding a narrative story to a juggling performance or for assisting in training novice jugglers.

While we take juggling as an example here, our techniques are generalizable to any scenario with objects moving with predictable motion paths. For example, one could imagine projected graphics on objects in free flight, free fall, objects that are swinging or bouncing, or objects with prescribed mechanical movement. As long as the motions are describable using physical laws (e.g., kinematics) we can predict the object's location and compensate for latency in projection.

4.2. *Scenario 2*: On-body Projection

Similarly to previously mentioned related work, such as LightGuide [5] and OmniTouch [3], we explore on-body projection for visual feedback. In contrast to the related work, our system specifically focuses on fast, dynamic motion. As in our juggling example, a person's hand movements can easily reach speeds that would result in projection misalignment due to system latency. Where our juggling scenario provides examples of inherently predictable features, such as the ball's flight path, this scenario requires the prediction of human motion, which is more subject to random variation and personalization.

While our focus here is on aligning projection with real-world objects, the human-motion prediction approaches we present could equally be applied to improve the responsiveness of interaction with virtual objects or, for example, in Kinect-enabled video games.

5. PREDICTABLE MOTION CATEGORIES

We split the motion prediction of objects observed by the pro-cam system into 3 categories: predictable, semi-predictable and unpredictable.

5.1. Predictable Motion

Predictable objects are those where, given a set of laws, their position at any point in time can be accurately determined. For example, due to the laws of physics, the projectile motion of our juggling balls falls into this category, as well as previously mentioned free fall, swinging, bouncing, locomotion, etc. While outside the scope of our scenarios (and more complex to predict), thermo-dynamics, magnetic fields and acoustics (for example) also fall within this category.

5.2. Semi-Predictable Motion

Semi-predictable motion includes objects whose motion typically follows a pattern or includes some repetition. Examples of these include a wide range of human motions, including walking, dancing given certain types of music (e.g., with a beat), or movement in sports [28].

Flash et al. show that human motion seeks to reduce 'jerk' (increase acceleration smoothness) in performance [29]. In its most basic form, this results in a linear motion between any two targets with acceleration following a bell curve. This is similar to the motion observed by Xia et al., when examining participant's movement towards a target on a touchscreen [19]. In more complex examples, research suggests that tennis player's moves can be anticipated (predicted) based on motion data, such as racquet position, shoulder rotation and lower body motion [30]. While not explicit, this implies the repetition of different tennis moves. Similarly, research highlights the cyclical nature of a juggler's motion [31]. Their hands move in an up-down (slightly elliptical) pattern – travelling upwards towards ball release and downwards during capture. Throughout this motion, the reduction of acceleration 'jerk' leads to a smooth movement.

Derived from these observations, we can begin to predict human motion based on individual performances. Following an initial performance, for example the interaction with a specific virtual target, we use a memory lookup table for prediction. We use current position and motion as input and an interpolated future predicted position as output. As the performance continues a more personalized and accurate model of motion can be developed. This estimation approach is explored later in this paper.

5.3. Unpredictable Motion

Motion that is random, such as a lay person's performance of a random task, is categorized as unpredictable. These provide us with no cues with which to reduce the effect of latency and are not addressed in our work.

6. METHODS FOR LATENCY REDUCTION

To begin to combat the effects of latency we explore predictable motion. We combined a Kinect camera and a DLP projector with an end-to-end system latency of 110ms as measured previously. We calibrated our projector to our Kinect camera, using a technique similar to that used in OmniTouch [3]. Of our two scenarios, the *Juggling Display* includes a predictable feature – the ballistic motion of the balls in free flight. Thus, we begin by exploring the ball's predictable ballistic trajectory. We use this motion estimation to predict the ball's location 110ms in to the future, projecting on to that location and thus reducing the effects of latency.

6.1. Predictable Motion – Kalman Filter with a Ballistic Model

Kalman filters are a popular approach to smoothing sensor data and estimating future data [13]. By fitting a Kalman filter with a ballistic motion model (in our case), the Kalman filter's prediction can take into account known physical behavior. In this instance, our projectile motion model is based on the following recurrence relation (using initial launch velocities and angles of release):

$$\boldsymbol{x}_t^* = \boldsymbol{x}_{t-1} + \boldsymbol{v}_{t-1}\Delta t + \frac{1}{2}\boldsymbol{a}_{t-1}\Delta t^2$$

where $\boldsymbol{x}_t^*$ is a prediction of the value of $\boldsymbol{x}_t$ given $\boldsymbol{x}_{t-1}$. Given observation $\boldsymbol{z}_t$ of the target's position, we update the estimated position, velocity and acceleration with:

$$\boldsymbol{x}_t = \boldsymbol{x}_{t-1} + \boldsymbol{k}_x * (\boldsymbol{z}_t - \boldsymbol{x}_t^*)$$
$$\boldsymbol{v}_t = \boldsymbol{v}_{t-1} + \boldsymbol{k}_v * (\boldsymbol{z}_t - \boldsymbol{x}_t^*)$$
$$\boldsymbol{a}_t = \boldsymbol{a}_{t-1} + \boldsymbol{k}_a * (\boldsymbol{z}_t - \boldsymbol{x}_t^*)$$

where Kalman gains $\boldsymbol{k}_x$, $\boldsymbol{k}_v$, $\boldsymbol{k}_a$ are computed according to [34] and relate the error in prediction of position, to changes in our estimates in position, velocity, and acceleration. For clarity, "*" denotes an element-wise operation while the rest are vector operations.

The Kalman filter incorporates our knowledge of sensor noise and recursively incorporates all previous observations to give us the principled means to set the value of Kalman gain given uncertainty in both prediction $\boldsymbol{x}_t^*$ and observation $\boldsymbol{z}_t$ [34]. For in-air motions, such as those of our juggling balls, we can assign very high certainty

to our acceleration estimate since the only force acting on the object is due to gravity (i.e., acceleration is constant at 9.81m/s^2). While a detailed explanation of the Kalman Filter is beyond the scope of this paper, we refer the reader to Welch and Bishop [34] for a good introduction.

In addition to this model, we specify low values for process noise, but relatively high uncertainty values for our observations due to quantization error (tracking through a rolling shutter and camera calibration errors). Observational data can be passed into the Kalman filter and as the filter's covariance and error estimates develop, increasingly accurate predictions can be made.

6.1.1. Application in the Juggling Display

We segment the balls from our depth image through an adaptive threshold and convert their position to real-world coordinates such that their size, location and velocity can be calculated at sub-pixel accuracy. Balls are tracked between frames using connected components. We use the Kalman filter's predictive step (with a variable time step) to estimate the future state of our system at any time; in our case, the future location of the juggling balls (similar to the approach in [22]). By predicting ahead according to our latency measures and using that prediction as a projected graphics location, we can project onto the real-world location of the ball (see Figure 1 and Figure 2). Without prediction, the projection only aligns with the ball at the zenith of the trajectory, equating to 14% of the ball's flight (as we show later in our results). Through prediction, we can align our projection with a greater portion of the ball's flight (Figure 4.)

However, due to the latency prediction step performed, further error is introduced by any interaction with the ball. Therefore, the projection continues passed the catch point for 110ms, or 3 further frames, introducing a new projection slip error.

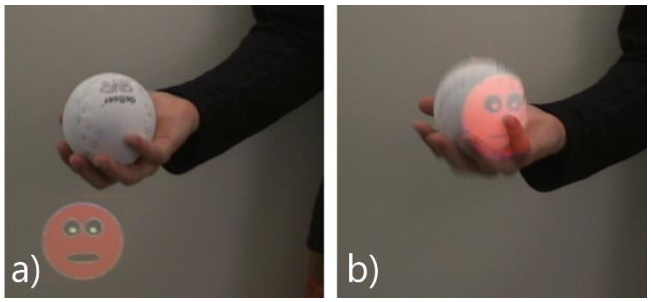

Figure 4. (A) Given a large predictive step, the projection can easily overshoot the juggler's hand. (B) By predicting the hand-ball intersection point, this overshoot can be avoided.

6.2. Semi-Predictable Motion – A Memory Lookup Model

When exploring predictable motion we used a Kalman filter fit with a known motion model. However, as motion becomes less predictable, we cannot provide an accurate relational model and thus look to other methods of prediction. One popular method of prediction in this case is the use of training data, such as used on touchscreens by Xia et al. [19]. However, as the scale of interaction increases, the variation in performance also increases, thus making a general training model less suitable.

Given a repetitive task, such as performed during a video game, when juggling or during sport (albeit repetitive over a longer window), we suggest a user's motion can be more accurately predicted based on their own previous performances (Figure 5). In contrast to a more generalized model, this approach takes into account the intricacies of personal performance, such as individual acceleration patterns, maximum reach and personal style. To this end, we present a memory lookup model. Through this model, movement details are stored as they are performed and then used to provide personalized training data for ongoing or subsequent performance. Given an action, we can perform a lookup into the memory model (based on speed, acceleration and position), locating previous examples of similar motion and interpolating between the subsequent stored data to provide an estimate of a future state. As performance continues, and further repetitions occur, this modelling technique increases in accuracy.

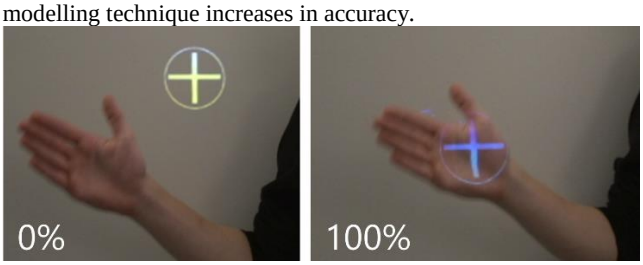

Figure 5. Image showing projection slip on fast human motion under no prediction and alignment under full latency prediction with memory lookup approach.

Both of our example scenarios, the Juggling Display and the on-body projection, include semi-predictable human motion and make use of our memory lookup model.

6.2.1. Application in the Juggling Display

In order to minimize error and maximize symmetry jugglers attempt to move as consistently as possible [32]. However, human error (such as angle, location and velocity of release) ensures that no two throws or catches are exactly the same [33]. As juggling motion repeats over a very short window, with hands moving in an ellipse to launch and catch balls typically more than once a second, we store the last 60 seconds of movement as provided through the skeleton tracking system. By creating a memory lookup model of the juggler's movement pattern, we predict future hand positions and thus determine the time and locations of catches. In turn, we can stop the projection at the point of catch and eliminate post-catch projection slip (as visible on the right of Figure 4).

6.2.2. Application for On-Body Projection

In our on-body scenario, where movement takes place over a greater number of patterns (moving towards 4 different target areas) and thus repeats less frequently, we adapt our memory store to store a greater amount of previous data. The player's hand positions are retrieved from the Kinect's skeleton data, converted to be relative to the base of the neck ('shoulder center' joint) and added to the end of a lookup list. We convert the hand positions from 'absolute' to 'relative to a central joint' so that previous positional data can be drawn upon as the player moves around their environment. When a new hand position arrives, speed and acceleration values are calculated from the last hand positions (the end of the lookup list). These values (location, speed and acceleration) are used as a lookup into the memory model. As the on-body scenario involves the player moving at fast speeds (> 3m/s), the Kinect's skeleton accuracy begins to degrade, resulting in reported hand values that fluctuate around the hand's true position (circa +/- 10cm.) This inaccuracy in sensing is taken into account when looking into the memory model. Our lookup process is as follows (and can be seen in Figure 4 below):

1. Locate all previously measured positions within a 10cm radius of our current hand position (relative to the center of the player's shoulders) (Figure 6: A and B).
2. Compare located positions motion with our lookup's motion, keeping only those travelling in a similar direction at a similar velocity (Figure 6: C).
3. Interpolate forward 110ms (our measured latency), from each located position, into the memory model to find the resultant position (Figure 6: C).
4. Find the average vector and calculate an average predicted location. (Figure 6: D).

It is worth noting, that we allow for 60 frames of data to be collected prior to using the lookup table such that some reference data exists (and then only use the table when a suitable match is found). Therefore, the initial 2 seconds of motion are subject to the same projection 'slip' as if no latency were being accounted for and the prediction accuracy increases as the user settles in to their motion and rhythm.

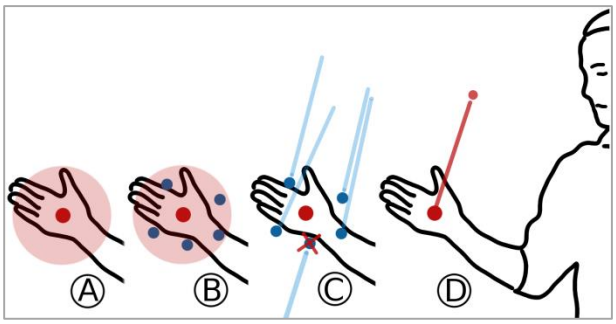

Figure 6. Memory table lookup steps. A) Determine hand position. B) Find near-located points (taking into account Kinect tracking error). C) Identify points with similar motion. D) Calculate average vector and use to predict location.

7. AUTOMATIC LATENCY TUNING

As an extension of our work on measuring and reducing latency in interactive systems, we developed both of our example scenarios to automatically measure and account for their own latency. We utilize the color camera to determine the location of the projected graphics in comparison to the center of the tracked object in any frame. (The projected graphics measure is converted to the depth camera space for accurate comparison.) Due to quantization error as a result of non-synchronization between our projector and camera, we use a recursive, latency traversal approach to finding the local minima for automatic latency tuning as opposed to a one-off minimization calculation based on two measurements.

We start with no prior assumption of latency and thus begin our measurement from 0ms. First, we step through the latencies in 33ms increments (the frame rate of the Kinect) until a local minima is found. We then refine our estimate using proportionally smaller latency changes, down to +/- 5ms. We found this to be sufficiently accurate given the minimum 16ms quantization due to projector frame rate and human perceivable accuracy changes below 5ms. We use this value in our motion prediction calculations. Due to the unlikely event of changing device configurations during use, we do not run the latency tuner continuously. However, we continue to monitor the Kinect and projector refresh rates, such that the update and predictive steps in our Kalman filters can be made with accurate time intervals.

The ability to automatically calculate and monitor the latency in pro-cam systems is an important step towards ensuring that interactive rates and a high rate of projection alignment can be maintained across different pro-cam setups and configurations.

8. SYSTEM EVALUATION

We assess the success of our pro-cam latency reduction techniques through empirical studies. Using our 2 example scenarios (predictable and semi-predictable motion), we provide an indication of how accurate our motion prediction and associated projection is to the object's actual position and at what speeds our system can accurately model human movement.

Across all of our studies we used a laptop PC with a core i5 processor (i5-3320M), 4GB of RAM and integrated Intel graphics. The system was built on Windows 7 and our rendering was through a Direct3D dedicated full-screen application which bypassed all operating system related compositing and rendering passes.

Throughout our evaluation, we compare our latency compensation with no latency compensation. We provide two different accuracy measurements. First, we capture an average projection distance offset using our automatic latency measurement approach. As this distance offset also illustrates calibration error. Second, we simultaneously use a frame counting technique to present a binary "projection aligned vs. projection missed measure" to clarify our results, which is computed by considering all the frames with an estimated >10% projection alignment. This additional measure is captured using an external camera (capture at 25fps, shutter at 1/60th second). In combination, these techniques provide an indication of our approach's success.

8.1. Study 1: Evaluation of our Predictable Motion Approach

In Study 1 we explore the success of our predictable motion Kalman filter approach through our *Juggling Display*. Three jugglers performed a 3 ball cascade ('standard' 3-ball juggling) for 2 minutes under no latency correction and full latency correction. The jugglers used softballs (9.7cm diameter) as they provide a clear projection surface and slightly larger image to track through the relatively low resolution Kinect depth camera. Our participants stood 2.5m from the pro-cam unit.

Alongside the automated offset measurements captured through the Kinect, a random 20 second segment from each 2 minute period was analyzed using frame counting (excluding the initial 10 seconds to allow for the development of a rhythm). The number of balls in flight and the number of balls under some projection were counted in all frames. In total, 1800 frames were analyzed per latency condition.

8.1.1. Results

Under no latency correction, we found that 14% of balls in flight are projected upon and that the average projection offset is 20.7cm. As previously suggested, we found that the majority of illumination occurs at the zenith of each ball's flight, where the speed approaches 0m/s. Under full latency compensation, we achieve 50% projection accuracy and an average projection offset of 7.47cm. This projection occurs from the zenith of the trajectory back to the juggler's hand.

While each individual ball is under projection for 50% of its travel time, due to the overlapping flight paths of the balls, 73% of our total captured frames contained some projection illumination. Without prediction, only 22% of frames contain any projection.

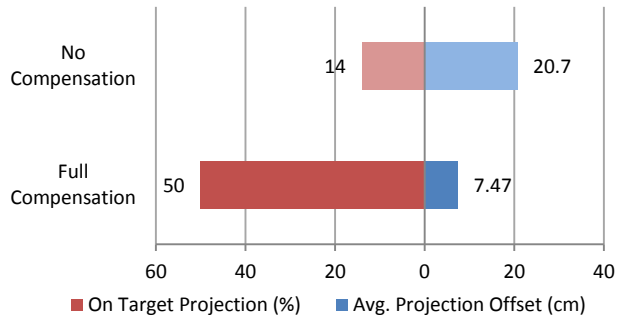

Figure 7. Average projection offset in cm and on target projection % under 'no' and 'full' latency compensation.

8.1.2. Analysis

Given a 30Hz camera and a 60Hz projector, we believe that our results approach the maximum projection alignment achievable by modelling predictable features of the activity.

To initialize the ball's flight in our Kalman filter we require two frames of data (to calculate velocity and angle of launch). An additional camera frame is required to smooth our observation data before we begin to predict the ball's location in the future (110ms per our latency measurement). Thus, initialization, smoothing and prediction equate to approximately 6 frames of Kinect data. Given an average ball flight duration of 0.5s or 15 Kinect frames, this results in between 6 and 7 frames, or 210ms, of missed projection. This leaves approximately 56% of the frames available for projection, as illustrated in Figure 8. At 50%, our achieved projection approaches this. We suggest that further accuracy is prevented by 2 features of our space.

First, a Kalman Filter's accuracy increases as more observations are incorporated. As our Kinect captures at 30Hz, we receive approximately 15 frames of data per-ball flight. From this data, we need to begin predicting future locations after only 3 observations, giving the filter little time to smooth our observation noise (as a result of the fast motion of the balls and rolling shutter of the camera). Furthermore, the quality of the filter's estimates decrease as the prediction window increases (predictions further into the future are made) [13], [22]. We need to estimate approximately 3.5 frames into the future and, even given our predictable motion model, we are subject to increased error in our Kalman filter's prediction.

Secondly, a ball traveling at 4m/s moves 13.2cm during a single camera frame (33ms @ 30Hz). Therefore, the quantization errors from the capture system alone contribute about 6.6cm to our error. Furthermore, at 2.5m from our pro-cam unit, each camera pixels measures 4.2x4.1mm and each projector pixel measures 2.7x3.25mm. Thus, our ball measures only 22 pixels across in camera space and 35 pixels across in projector space. This small size introduces further error into both our tracking and measurement systems.

So although our techniques enable us to increase projection accuracy on objects in free flight, we are unable to achieve complete projection given our hardware configuration and prediction techniques.

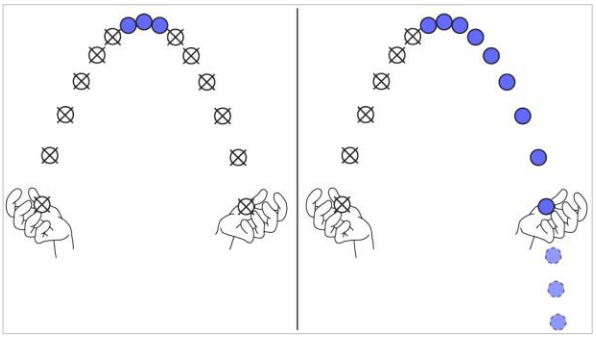

Figure 8. An illustration of the portion of flight where projection occurs given no latency compensation (left) and full latency compensation through predictable motion (right).

8.2. Study 2: Evaluation of Semi-Predictable Motion Correction – Memory table lookup Approach

In Study 2 we evaluate the success of our memory table's ability to predict fast human motion. Three participants performed up-down and circular hand movements, pivoting at the elbow, at 1m/s and 1.5m/s in an up-down motion and 1.5m/s in a circular motion. A 10cm diameter graphic was projected onto the hand's location, as provided by the Kinect's skeletal tracker. The system provided visual cues as to what range of movement to perform. In order to allow a personalized approach, the participants were provided with a target movement speed but no assistive timing feedback was provided. Each participant performed each movement for 15s. The participants were all right-handed and performed the movements with their right hands. As in our previous study, we provide both a projection offset and binary 'on target' measure.

Although in this instance we do have access to known target locations, as we are interested in our prediction's applicability in situations where no additional knowledge is available, we do not use target location to influence our prediction.

The participants completed the study using both our memory model and a Kalman filter model. A Kalman filter could also be used for prediction here, but is likely less accurate than our approach due to the fast changes in acceleration and direction, a large predictive step and our inability to provide an accurate motion model. We would expect the Kalman filter to track effectively during linear motion, but overshoot upon dynamic changes in direction. We configured a Kalman Filter to track and update based on both the velocity and acceleration of the juggler's hands. We assigned high uncertainty to our estimate of acceleration, while assigning a strong weighting to our sensor data, such that the filter performs responsively. It may be possible to further improve the Kalman filter with a human kinematic motion model, but this is outside the scope of our paper.

8.2.1. Results

Our overall results show (Figure 9) that 69% of the projected frames fall on-target with the memory model at full latency compensation, with an average projection offset of 7.3cm. Without compensation, only 26.3% of frames fall on target (average offset 12.8cm). In comparison, the Kalman Filter achieved accuracy of only 16.45% at full latency compensation, with an offset of 17.9cm.

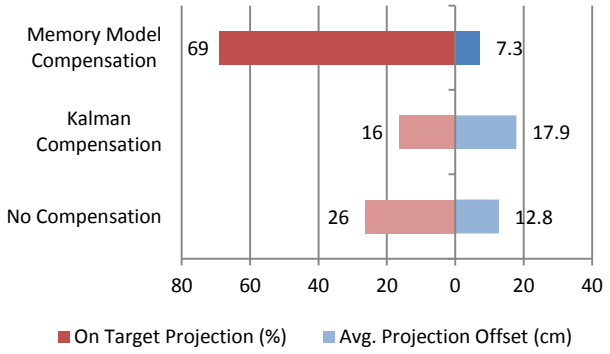

Figure 9. The projection accuracy comparison between the memory model and the Kalman Filter approach. Values show the percentage of frames that landed on-target during semi-predictable portion of the ball motion.

8.2.2. Analysis

At 69% we again believe our results are near the best projection accuracy achievable within our system. For our target position of the hand we use the value reported by Kinect's skeletal tracking. Kinect skeletal tracking filters each joint value, which makes it likely that the accuracy degrades at higher speed. Indeed, at 1m/s+, we are approaching the limits of the Kinect's skeleton tracking. While it is possible to modify the amount of filtering, we chose to keep the default settings in order to make our results easily comparable to other pro-cam systems using Kinect. Anecdotally, we identified approximately 10cm of noise in the Kinect's reporting of our skeletal location at speeds greater than 1m/s.

Interestingly, the Kalman filter results are relatively linear across both latency compensations. When predicting ahead during linear motion, the Kalman filter provides increasingly accurate location predictions with increases in latency compensation. However, with an increased latency prediction, the changes in direction cause a greater projection error, compensating any gain experienced through the linear phase.

9. DISCUSSION

The results of our studies support our approach of utilizing software based techniques to negate the effect of latency in high motion pro-cam systems. We increased projection accuracy on objects in free flight, from 14% of the ball's flight to 50% in our juggling scenario, drastically reducing the average projection offset from 20.7cm to 7.47cm. We also increased on-body projection by 43%.

We suggest that, given the latencies inherent within a pro-cam setup of this kind (with a 30Hz camera and a 60Hz projector) our results approach the maximum achievable projection accuracy. In our system, a small data sample size (15 frames on average per throw) and noisy data (whether from the Kinect skeletal tracker or rolling shutter effects), combined with a large predictive step (circa 100ms) make it impossible to achieve perfect projection accuracy. Furthermore, at these speeds, the human observation of the balls is also subject to motion blur. For this reason, while the projection time is relatively short, the juggling display is compelling to observe. We encourage the reader to see the accompanying video of our system in action to see the results of our approaches.

Ultimately, reducing latency in a system requires both hardware and software optimizations. However, in this paper we specifically chose to focus on software-only approaches to demonstrate that, even with relatively high-latency pro-cam configurations, one can drastically improve the system latency by considering the predictability of features of the use case. Thus our approach offers a solution for enabling compelling fast-motion pro-cam systems with off the shelf hardware.

10. CONCLUSION

In this paper, we explored software approaches for reducing the effects of latency in a pro-cam system, with the intention of enabling wider exploration of dynamic pro-cam settings. Our results show that our approaches of addressing latency are promising and we encourage the reader to watch our video to see the system in action. We present a 36% increase in projection time on individual balls, a 51% increase in number of total frames including projection and a 43% increase in human motion predictability. We specifically avoided making any adjustments to our hardware and used typical off-the-shelf devices with relatively high latencies.

While these are not the only solutions to combat pro-cam system latency, we believe that our approaches add valuable solutions to the palette of options that should be considered when designing pro-cam systems. In addition, the approaches presented here are applicable to many other interactive systems that deal with latency due to object motion. We hope that our latency compensation solutions enable wider adoption of pro-cam systems for highly interactive fast-moving scenarios.

11. REFERENCES

[1] B. Jones, R. Sodhi, M. Murdock, R. Mehra, H. Benko, A. Wilson, E. Ofek, B. MacIntyre, N. Raghuvanshi, and L. Shapira, "RoomAlive: Magical Experiences Enabled by Scalable, Adaptive Projector-camera Units," in *Proc. UIST*, 2014.

[2] H. Benko, A. D. Wilson, F. Zannier, and H. Benko, "Dyadic Projected Spatial Augmented Reality," in *Proc. UIST*, 2014.

[3] C. Harrison, H. Benko, and A. D. Wilson, "OmniTouch: Wearable Multitouch Interaction Everywhere," in *Proc. UIST*, 2011.

[4] S. A. Seah, D. Martinez Plasencia, P. D. Bennett, A. Karnik, V. S. Otrocol, J. Knibbe, A. Cockburn, and S. Subramanian, "SensaBubble: A Chrono-sensory Mid-air Display of Sight and Smell," in *Proc. CHI*, 2014.

[5] R. Sodhi, H. Benko, and A. Wilson, "LightGuide: Projected Visualizations for Hand Movement Guidance," in *Proc. CHI*, 12.

[6] F. Anderson, T. Grossman, J. Matejka, and G. Fitzmaurice, "YouMove: Enhancing Movement Training with an Augmented Reality Mirror," in *Proc. UIST*, 2013.

[7] M. Löchtefeld, S. Gehring, R. Jung, and A. Krüger, "guitAR: Supporting Guitar Learning Through Mobile Projection," in *Proc. CHI EA*, 2011.

[8] A. Barnett, "The Dancing Body As a Screen: Synchronizing Projected Motion Graphics Onto the Human Form in Contemporary Dance," *Comput Entertain*, vol. 7, no. 1, pp. 5:1–5:32, Feb. 2009.

[9] A. Ng, J. Lepinski, D. Wigdor, S. Sanders, and P. Dietz, "Designing for Low-latency Direct-touch Input," in *Proc. UIST*, 2012.

[10] H. Yamaguchi, and H. Koike, "LumoSpheres: Real-time Image Projection based on real-time tracking of flying objects," presented at the WISS, 2013.

[11] H. Benko, R. Jota, and A. Wilson, "MirageTable: Freehand Interaction on a Projected Augmented Reality Tabletop," in *Proc. CHI*, 2012.

[12] R. S. Allison, L. R. Harris, M. Jenkin, U. Jasiobedzka, and J. E. Zacher, "Tolerance of temporal delay in virtual environments," in *Proc. IEEE VR*, 2001.

[13] J. Liang, C. Shaw, and M. Green, "On Temporal-spatial Realism in the Virtual Reality Environment," in *Proc. UIST*, 1991.

[14] C. Ware and R. Balakrishnan, "Reaching for Objects in VR Displays: Lag and Frame Rate," *ACM Trans CHI*, vol. 1, no. 4, pp. 331–356, Dec. 1994.

[15] G. Papadakis, K. Mania, and E. Koutroulis, "A System to Measure, Control and Minimize End-to-end Head Tracking Latency in Immersive Simulations," in *Proc. VRCAI*, 2011.

[16] K. Okumura, H. Oku, and M. Ishikawa, "Lumipen: Projection-Based Mixed Reality for Dynamic Objects," in *Proc. IEEE ICME*, 2012.

[17] K. Okumura, H. Oku, and M. Ishikawa, "High-speed gaze controller for millisecond-order pan/tilt camera," in *Proc. IEEE ICRA*, 2011.

[18] R. Jota, A. Ng, P. Dietz, and D. Wigdor, "How Fast is Fast Enough?: A Study of the Effects of Latency in Direct-touch Pointing Tasks," in *Proc. CHI*, 2013.

[19] H. Xia, R. Jota, B. McCanny, Z. Yu, C. Forlines, K. Singh, and D. Wigdor, "Zero-latency Tapping: Using Hover Information to Predict Touch Locations and Eliminate Touchdown Latency," in *Proc. UIST*, 2014.

[20] K. Kitani, K. Horita, and H. Koike, "BallCam!: Dynamic View Synthesis from Spinning Cameras," in *Proc. UIST Adjunct*, 2012.

[21] L. Fairfax and F. Fresconi, "Position Estimation for Projectiles Using Low-Cost Sensors and Flight Dynamics," *J. Aerosp. Eng.*, vol. 27, no. 3, pp. 611–620, 2014.

[22] M. Friedmann, T. Starner, and A. Pentland, "Device Synchronization Using an Optimal Linear Filter," in *Proc. I3D*, 92.

[23] A. Steed, "A Simple Method for Estimating the Latency of Interactive, Real-time Graphics Simulations," in *Proc. VRST*, '08,

[24] M. R. Mine, "Characterization of End-to-End Delays in Head-Mounted Display Systems," University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, 1993.

[25] Chow, J., Ang., K., Lichti, D. and Teskey, W., "Performance Analysis of a Low-cost Triangulation-based 3D Camera: Microsoft Kinect System.," International Archives of P, RS and SIS, 2012, vol. 39, pp. 175–180.

[26] K. Khoshelham, "Accuracy analysis of kinect depth data," in *ISPRS workshop laser scanning*, Calgary Canada, 2011, vol. 38.

[27] M. Asad and C. Abhayaratne, "Kinect depth stream pre-processing for hand gesture recognition," in *Proc. IEEE ICIP*, 2013,

[28] "Coupled Oscillators and Biological Synchronization." [Online]. Available: http://www.scientificamerican.com/article/coupled-oscillators-and-biological/. [Accessed: 05-Nov-2014].

[29] T. Flash and N. Hogan, "The coordination of arm movements: an experimentally confirmed mathematical model," *J. Neurosci.*, vol. 5, no. 7, pp. 1688–1703, Jul. 1985.

[30] J. Shim, G. Les Carlton, and Y.-H. Kwon, "Perception of kinematic characteristics of tennis strokes for anticipating stroke type and direction," *Res. Q. Exerc. Sport*, vol. 77, no. 3, pp. 326–339, Sep. 2006.

[31] A. D. A A Post, "Principal components in three-ball cascade juggling.," *Biol. Cybern.*, vol. 82, no. 2, pp. 143–52, 2000.

[32] R. Huys and P. J. Beek, "The coupling between point-of-gaze and ball movements in three-ball cascade juggling: the effects of expertise, pattern and tempo," *J. Sports Sci.*, vol. 20, no. 3, pp. 171–186, Mar. 2002.

[33] "The Science of Juggling." http://www.scientificamerican.com/ article/the-science-of-juggling/.

[34] G. Welch and G. Bishop. Introduction to the Kalman Filter. *Technical Report.* University of North Carolina at Chapel Hill. TR 95-041. 2004.