Augmenting Information Flow for Visual Privacy

by

Ian Spiro

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

New York University

September 2013

Professor Christoph Bregler

Acknowledgements

First and foremost, I would like to thank my partner Molly Tanenbaum for her love, patience, good cheer, and support. I'm excited for our next chapter and looking forward to the day that we can enjoy spiced lamb shanks in a hot tub.

Thanks to Chris Bregler, my research advisor and patron. He encouraged me to explore new intellectual paths and brought me along for some wild rides.

Thanks to Helen Nissenbaum for leading the Privacy Research Group and for the great discussions in which we managed to bridge the interdisciplinary abyss.

Thanks to my family, Mom, Dad, and Miranda.

Thanks to the Motion Chain alpha users who tested and invented charades: Phil Dhingra, Raphael Holmes, Kiefer Katovich, Carrie Kemper, Doug Kenter, Devon Sherman.

I met some great people at NYU who made the experience considerably more enjoyable: Clement Farabet, Yotam Gingold, Eric Hielscher, Adrian Secord, Kirill Smolskiy, Murphy Stein, Graham Taylor, Matthew Tierney, and George Williams.

Preface

The work presented in chapter 2 is based upon two previously published papers. The first paper appeared in Computer Vision and Pattern Recognition Workshops (CVPRW 2010) and was co-authored with Graham Taylor, George Williams, and Christoph Bregler [80]. The second paper was presented at Collective Intelligence 2012 [79]. This work was co-authored with Thomas Huston and Christoph Bregler.

Chapter 4 is based on a paper originally published in the extended abstracts of the conference on Human Factors in Computing Systems (CHI 2012) [78].

Chapter 5 is based on a paper that will be published in the proceeding of the ACM Conference on Online Social Networks (COSN 2013). This paper was co-authored with Matthew Tierney, Christoph Bregler, and Lakshminarayanan Subramanian [88].

Abstract

In the Information Age, visual media take on powerful new forms. Photographs that were once printed on paper and stored in physical albums now exist as digital files. With the rise of social media, photo data has moved to *the cloud* for rapid dissemination. The upside can be measured in terms of increased efficiency, greater reach, or reduced printing costs. But there is a downside that is harder to quantify: the risk of private photos or videos leaking inappropriately. Human imagery is potentially sensitive, revealing private details of a person's body, lifestyle, activities, and more. Images create visceral responses and have the potential to permanently damage a person's reputation.

We employ the theory of *contextual integrity* to explore certain privacy aspects of transmitting the human visual form. In response to privacy threats from new sociotechnical systems, we develop practical solutions that have the potential to restore balance. The main work is a set of client-side, technical interventions that can be used to alter information flows and provide features to support visual privacy. In the first approach, we use crowdsourcing to extract specific, useful human signal from video to decouple it from bundled identity information. The second approach is an attempt to achieve similar ends with pure software. Instead of using information workers, we develop a series of filters that alter video to hide identity information while still revealing motion signal. The final approach is an attempt to control the recipients of photo data by encoding data in the visual channel. The software completely protects data from third-parties who lack proper credentials and maintains data integrity by exploiting the visual coherence of uploaded images, even in the face of JPEG compression. The software offers end-to-end encryption that is compatible with existing social media applications.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Our society has witnessed the increasing prevalance of visual recording devices in both outdoor and indoor spaces. Urban environments are subject to surveillance for security purposes while digital cameras have become standard in nearly all consumer laptops and mobile phones. This ad-hoc network of visual sensors has the potential for unprecedented reach into our lives. In parallel, we see a significant cultural shift toward the adoption of software with *social* purposes. Unlike much of earlier software that focused on workplace productivity, social media fills a role of entertainment or personal expression. This online software has become embedded deeply in our world and has an influence on social structures that cannot be ignored.

Social networks provide services free of charge to users who grant broad consent when they agree to lengthy terms of use, terms that are subject to constant modification. This gives social networks great leeway in what they can do with user data. Users rarely understand the detailed legal aspects of their transactions with a social network nor do they typically even read the terms of use [5]. Privacy

advocates see many troubling possibilities that arise in such a sociotechnical landscape. Access control to sensitive personal data is implemented by social networks and users are asked to trust that the companies will get it right. But if a social network is hacked or experiences other technical problems, private data can leak to the world. Many examples can be found of individuals losing jobs or social standing when private information flows inappropriately, either due to software bugs or user error. A poignant example occurred when Mark Zuckerberg, CEO of Facebook, had personal photos he had marked as private leak to the entire Internet because of a temporary system glitch [22]. Visual imagery of humans is especially sensitive and subject to privacy concerns in light of social software. We say *seeing is believing*. Images convey a huge range of human experience, and unfortunately, that includes images that can irrevocably damage a person's reputation.

In 1986, Langdon Winner asked, "Do artifacts have politics?" That is, can an artifact or tool (or website) systematically favor certain human values over others? He and many subsequent scholars believe the answer is yes. An artifact has politics that reflect the people who created it. Silicon Valley entrepreneurs give users free services as a way to sell advertisements. Their priority is to learn enough about their users to better serve their advertising clients. A response to a person skeptical of social media might be: "If you don't like it, don't use it." And while the truly paranoid may avoid social media altogether, this is unrealistic for the vast majority of people. Social media has become intertwined in the basic fabric of society and to reject it could mean to ostracize oneself. Technology, for much of the population, happens. When browsing a social network, refreshing the web browser can load a new version of the site, creating a change in the underlying logic that is inscrutable to the user but with real-world consequences.

People unconsciously adapt to the ever-shifting sociotechnical environment and corresponding corporate policies. Though the underlying technology may appear straightforward and benign, we cannot simply accept or ignore it. Social, moral, and political ramifications of new technology necessitate discussion, research or even changes in policy.

1.1 Project History

This thesis got its start not as a work of privacy scholarship but through our attempts to use computer vision in the study of human motion. The Movement Lab studies motion in a great variety of forms using both studio-based motion capture and software-based approaches. Traditional motion capture records the position of human subjects' skeletons over time and can be used for future playback, artistic applications, or for scientific study. A standard recording session takes place inside a pre-calibrated capture volume with many fixed, high-speed cameras. The human subject must wear a tight-fitting suit with retroflective markers placed according to a marker set. This is not a lightweight process and is often too cumbersome for studying motions *in the wild*, for example, at sporting events or dance performances. This limitation has motivated numerous attempts at *markerless* motion capture, in which motion capture data is derived from regular video footage, typically after the fact. Though many attempts have been made to fully automate such a process, none are generally accurate enough for practical applications like animating a 3D character. In many cases, human tracking artists are employed to either clean up the motion data or annotate it from scratch, also known as rotoscoping [3]. With the advent of crowdsourcing, we set out to build a crowd-powered

system, a generic pipeline to perform markerless motion capture with the goal that it be cheap, fast, and accurate [76]. We succeeded in getting online workers, acting as a pool of tracking artists, to annotate videos of moving people to extract 2D motions. Animated visualizations of 2D motion capture were immediately compelling. Though greatly impoverished in comparison to the original video pixels, just 13 moving dots could strongly suggest a living human form. While much of our research has focused on the movement of professional athletes or performers, we are also interested in more pedestrian movements. On the top floor of our building on Broadway in Manhattan, we may observe people walking past at all hours of the day and night. What if we make recordings of pedestrians and use the same pipeline to extract motion data? With multiple camera angles, it would be possible to reconstruct 3D motion capture data [63].

This project, however, started to raise unexpected questions. Something in the act of processing street video with an algorithm—a human-powered algorithm no less—felt potentially questionable. If body language could be automatically extracted, it might eventually be used to guess at the mental disposition of a human subject. And a variety of papers have shown motion capture data has potential as a *soft-biometric* [94], that is, personally identifying like a fingerprint with imperfect but probabalistic accuracy. Using such systems might be reasonable in academic contexts, but what of commercial or government appropriations of the technology? On the other hand, what if extracted motion signal could *help* to anonymize video footage? By extracting and transmitting only motion capture, we retain useful signal while arugably helping to protect the identities of the humans in question. In either case, we see that a vision-based technology alters the flow of information and may raise social or moral questions. Our native discipline of Computer Science

lacks the vocabulary to answer such questions so we turn to the academic area of *values in design*. With a theoretical framework called *contextual integrity*, we are able to better understand and analyze the ramifications of augmenting human video data.

1.2 Contextual Integrity

Contextual integrity is a theory developed by privacy researcher Helen Nissenbaum [55]. Many previous attempts to define privacy employed dichotomies. For example, one approach defines privacy based on the type of the information being exchanged: medical history is private but property ownership records are not. Others have tried to define privacy based on physical location: events inside the home are private and events that happen on the street are public. But these simple approaches have failed in the face of complex, real-world scenarios. In contrast to these dichotomies, Nissenbaum proposes contextual integrity as a way to study how information flows in different social scenarios and to understand what exactly is at stake when a new piece of information technology is introduced. Privacy is defined as the *appropriate flow of information.* To understand what is appropriate and to further describe the theory, some definitions are necessary.

1.2.1 Definitions

Context refers to a specific type of social scenario. *Actors* refer to the people involved and includes the *receiver* of information, the *sender* of information, and the human *subject* of the information, which might be the same as the sender. *Attribute* is the type and nature of the information being conveyed. The *transmis-*

sion principle describes the information flow, a social template that could apply in a variety of scenarios. For example, a transmission principle of *reciprocity* implies that two actors are both senders and receivers with a symmetric information flow. Contextual integrity is defined as the appropriate flow of information via some means of transmission between actors in a specific social context. Contextual integrity can be violated by new systems if there is a disruption in the flow of information. This disruption stems from a technology-driven change in the actors, attributes, or transmission principles in a context. After a potential violation is flagged because of such a change, it doesn't necessarily mean the new technology is bad. We have to further consider how the technology supports or impinges on context-specific goals as well as more general social, political, and ethical values. (See Chapter 8: *Breaking Rules for Good* in [55] for a detailed explanation.)

1.2.2 Example

In a medical caregiving context, a patient provides information about his body to a doctor. The patient is both the subject and the sender of health-attribute information to a doctor, the recipient. The means of transmission could be described as *physician-patient privilege*, a form of confidentiality, dictated by the doctors' code of ethics, a variety of laws, and the prevailing norms of society. This flow of information from patient to doctor is considered normal and appropriate. Now imagine a new piece of technology, a website for doctors and patients to collaboratively track the patient's health. If the website made the patient's health information accessible to friends, it would break contextual integrity. The recipient has expanded from the doctor to include the subject's friends. If this increased sharing of personal information with friends does not aid the subject's

or the public's health (goals of the health care context), then the change should be rejected.

1.3 Contextual Integrity of Motion Capture

Returning to the street recording scenario, we see that augmenting a recording of pedestrians with motion capture does alter informational flow. One obvious change is that the recipient of street footage data has expanded to include remote information workers. But more significantly, the new technology expands the attributes of the imagery data. Regular video data has been augmented to include an extracted motion signal, the subsequent derivatives of which are not fully understood. Determining the actual moral significance of the technology depends completely on the scenario in which it is employed. We might conclude that for certain security purposes, it is reasonable. But an individual using the system to obtain a commercial advantage over someone else (say, determining mental state to target a sales pitch) could be deemed unacceptable.

1.4 Motion Capture for Visual Obscurity

Though our research effort to extract motion capture data from video has possible implications for privacy, we can also look for positive applications of the same technology. Consider a scenario in which a surveillance application brings a clear social benefit. For example, a city planner wishing to optimize an urban environment studies the movement of pedestrians within a space. The task benefits from the ability to track human motion but does not require knowing any observed subject's identity. By processing surveillance imagery with the foregoing motion

capture pipeline and *throwing out* original video data, pedestrians can still be studied but with a significantly lower risk of being identified.

1.5 Motion Visualization

In parallel to our human-powered effort, there have been many attempts at purely algorithmic techniques for studying human motion. One successful computer-vision effort was the processing of video to determine the relative gestural similarity of pairs of human subjects [97]. One happy side-effect of this approach appeared during the debugging phase of development. By playing back animations of estimated optical flow vectors, visualized as a set of moving dots over time much like traditional motion capture playback, the movement is visualized. In regular motion capture, a fixed marker set identifies joint and limb configuration over time, while the debugging technique produced a grid of displacement vectors suggesting motion. We could see the approximate movement of subjects (head nods or hand waves, for example) even if the subject's skeleton was not explicitly tracked. We further iterated on this approach at visualization by computing sparse optical flow for extracted feature points. This led to a filter known as *greendot*. (The color choice is historical.) As before, the removal of original image pixels appears to reduce or obscure the human identity of the subject being visualized.

1.6 Interventions

1.6.1 Alter Attributes

These methods of motion visualization allow us to separate human motion from human identity, either through explicit tracking of a human skeleton or through algorithmic visualization of optical flow. In the case of public surveillance, it would take a policy change to convince security operations to implement such a filter. But for social media, where users typically use their own hardware to interface with the system, the methods can be directly employed as an intervention to augment informational flow, in an attempt to maintain or restore contextual integrity. Assuming reasonably open hardware and APIs, a user device can be modified to upload not original imagery but a processed version instead. This is essentially what applications like Instagram do. These apps upload filtered versions of photos or videos to a social network for aesthetic purposes. The greendot treatment is a visual effect, yet it also helps the end user limit transmitted video attributes.

1.6.2 Alter Recipients

The final section of this thesis attempts to aid visual privacy by altering not the information attributes but the recipients in a social media context. As contextual integrity explains, control over the recipients of personal information is a key aspect of privacy. This control has been threatened in online social networks, software that is designed in part to encourage sharing and expansion of 'friend' lists, and in some cases, default settings that make shared media world-readable. In the age of social media, friction is reduced so thoroughly that one is a single click from sharing an image with any online friend or the general public of the Internet.

To aid contextual integrity, we developed an end-to-end encryption tool that allows a user to upload encrypted images to a social network. This creates a separate channel for security completely independent of a social network, implemented with an encryption standard and user-generated passwords. By taking control of the cleartext form of images at the pixel level, as opposed to relying on social network privacy controls, the user takes greater control over the recipients of personal imagery.

1.7 Organization

Chapter 2 addresses the technique of extraction to separate motion and identity signals. We employ information workers to collaboratively rotoscope videos of human subjects using a keyframe tool and an explicit marker set. The resulting motion capture can be played back as a cartoon to hide all original image texture. This method is not real-time but achieves a strong level of anonymization by hiding all data except underlying motion. The capture is useful for motion applications, whether artistic, scientific, or social. We believe the user's visual identifiability will be reduced in the general case.

Chapter 3 focuses on obfuscation. Instead of using humans to extract motion, we process video with filters in an attempt to separate human motion and human identity. Consider as an example the use of pixelation in broadcast media to allow witnesses to communicate without being identified. We set out to compare a variety of filters for their ability to obscure identity while still transmitting body language or gesture. To this end, we first produced a novel corpus of gesture footage using a game with a purpose called *Motion Chain*. With the resulting corpus and

crowdsourced perceptual experiments, we assessed the effects of various filters on motion and identity transmission. We show that the *greendot* filter in particular offers useful properties. A user who applies such a filter locally is still able to use social media with a webcam or video upload but the visual information transmitted will be diminished to increase likelyhood of anonymity.

Chapter 4 describes the aforementioned game with a purpose, Motion Chain, not a technical intervention but a means for collecting human gestures. With a corpus of human body language, we can perform both machine learning and human (perceptual) experiments.

Chapter 5 concerns encryption and demonstrates a rigorous way to secure private visual information. While extraction and obfuscation techniques dramatically alter input signal, they still produce something with an apparent cleartext form. In this chapter, we offer security such that no private visual information is conveyed whatsoever. Through a set of tools called *Cryptagram*, photo data is transformed such that it requires a password to view. Encrypted data is exported as an image that is compatible with existing social media. Users upload these transformed images as they would regular JPEGs but they appear scrambled. Only users who install the decoding software and have the correct password will be able to view original photos. By using a lightweight browser extension, images are decoded *in-place*. Encryption is achieved on top of an existing social media user interface. Users can combine the password-augmented image format with existing social network privacy features for even greater security.

Chapter 2

Markerless Motion Capture

2.1 Overview

This work uses crowdsourcing to extract gesture data and motion capture from video recordings of human subjects. The data is obtained by a pool of online information workers (i.e. crowdsourcing) who click repeatedly to indicate body configurations in the frames of a video, resulting in a model of 2D structure over time. We exploit redundancies in video data in such a way that workers' efforts can be multiplied in effect. In the end, a fraction of frames need to be annotated by hand, but we can still achieve complete coverage of all video frames. This is achieved with a novel user interface and automatic techniques such as template tracking and affinity propagation clustering. We discuss techniques to optimize the tracking task and strategies for maximizing accuracy and efficiency. We show visualizations of a variety of motions captured with our pipeline then apply reconstruction techniques to derive 3D structure.

2.2 Introduction

In traditional motion capture, a capture volume is established with several high speed cameras that track retroreflective dots on actors' clothes. This is acceptable for high-budget animation projects where entire scenes will be rendered from scratch with motion capture applied to animated characters. But the equipment is expensive and in many cases it simply isn't possible to fit an activity of interest into a capture volume. There is increasing interest in motion capturing sports activities, either for use in games, movies, or sports medicine. Sports are notoriously difficult to motion capture, as they tend to require large capture volumes and the motion capture suits may interfere with the motions in question. A broader limitation of traditional motion capture is that it must be arranged for in advance and many of the most interesting motions to study are found in history. We have vast archives of human motion recorded in video but without explicit information of the subject's joint configuration over time, as we get in motion capture. Automatic video-based tracking without motion capture markers is a very active field in Computer Vision, but so far no working general solutions have been proposed, as we discuss in the Vision-based Tracking section.

With this motivation, we built a system for capturing motion from video. We developed a tool for users to perform annotation of arbitrary sets of points in a video. The interface is deployable in a web browser so we can pay users of Amazon Mechanical Turk (AMT) to complete the complex tracking task. With the aid of keyframe interpolation, a person can perform our task efficiently. We also deploy tasks redundantly to exploit the powerful effect of averaging in a crowd.

We focus our efforts on acquiring annotations of human motion features such as hand position and complex 3D skeleton configurations. Sometimes called match-

moving, this is a typical task in high-end visual effects studios and an entire suite of internal and external tools [38, 39, 9] are used by professionally trained "tracking-artists." Academic researchers also have a need for this and high-end annotation tools [43, 65] have been developed and used by gesture annotation experts previously. In both cases, a large budget is usually necessary to annotate a small set of videos.

In support of our research, we generally attempt to build systems that can extract patterns and statistics from large video databases, and can be used for various tasks in gesture analysis, to aid social and behavioral sciences, and for general search and visualization tools centered around understanding human motion. We previously developed a prototype system that can learn statistical models of individual "motion style" for subjects that are engaged in speaking actions. We have successfully applied this to a variety of tasks, including automated identification of personal speaking style, and to more general body motion styles [98]. That specific system uses no explicit representation of hand motion, or gesture types. This chapter describes our efforts toward building a richer training database that contains detailed labels of hand motion and pose information.

The challenge is to transform a complex, labor intensive task such that it can be broken up into smaller units and completed by a variety of users without expert knowledge of the domain. We need to avoid treating every individual frame as a task, as our database has on the order of millions of frames and this would be cost-prohibitive by any conservative estimate. We employ a staggered approach that alternates between human input and clustering techniques, such as affinity propagation [27] and pattern tracking techniques [46]. The UI attempts to replicate only the bare essentials of high-end tools. In some cases we introduce new ways of

Figure 2.1: An example annotation from the pilot dataset: Vladimir Zhirinovsky making an open-palmed, symmetrical, double-handed loop.

doing the task. This is a non-trivial user interface design challenge, targeting the interface to crowdsourced workers such that it can be learned quickly with minimal training, but provides us with quality data.

This chapter details the system design, which overall is a set of vision and machine learning tools running on a server, and Adobe Flash on the front-end, facing workers who have been sourced through Amazon Mechanical Turk. We also describe the design and deployment of HITs. This AMT-based system complements our other research efforts that use fully automatic vision and statistical learning approaches to extract similarly meaningful representations from video data [98].

2.3 Related Work

2.3.1 Annotation Tools

The task of tracking the position of features or objects in a video, and "matching" the 3D pose and configuration of rigid and articulated objects in 3D is, in the visual effects industry, known as "match-moving" or "rotoscoping." Methods date back over 100 years when animators traced film on light tables to produce animations. Most recent high-end tools are based on supplying the user with an interface that is similar to 3D key-frame animation (such as seen in Autodesk Maya and other tools). The user can "scrub" the video back and forth, and can click on locations to set key-frames. The frames in between are either interpolated or semi-automatically tracked with general pattern trackers ([38, 39] to name a few). Most recent advances that are closest to our philosophy are techniques that blend hand annotations and automatic tracking and model estimation in an interactive way [11, 9]. Another community, which includes gesture and multi-modal communication researchers, use a different set of annotation tools. The most popular ones are ANVIL [43] and MacVisSTA [65]. Both tools are more targeted for time-based annotation of text tags, but have some capability of annotating spatial information. Neither tool permits annotation of 3D configuration information. All tools discussed so far have in common a high level of complexity that requires a user to undergo some training. In some cases the level of training needed to use these tools is extensive. This is not possible for AMT users: they need to understand how to use an annotation tool in a very limited time. Another problem with the high-end tools is that they are generally not platform independent. For AMT we don't want to require that users have a specific operating system. Therefore we

employ a web-based interface primarily written in Flash.

LabelMe [66] and the Web Annotation Toolkit [37] provide web-based toolboxes for image annotations. And most recently a video extension [100] has been reported. In [77], they build a system that obtains coarse pose information for single frames but without video support. Another project [93] is specifically geared toward video of basketball but the resulting data is a set of human-size bounding boxes with no details of body pose. For our specific domain, these toolboxes do not provide the necessary functionality, since our annotations generally require handling of non-rigid objects and nuanced, high-speed motions.

2.3.2 Vision-based Tracking

General vision-based human body tracking has gained increased attention in recent years [10, 20, 23, 74, 75, 52, 62] but typically breaks on complex motions, motion blur, low-resolution imagery, noisy backgrounds, plus many other conditions that are present in most standard real-world videos. It is beyond the scope of this chapter to review all related tracking techniques and we refer to [26] for a survey. Fully automatic tracking of gestures is not solved yet for the general case. Part of our research agenda is to build a training database that will be used to develop and further improve such automated gesture tracking systems.

2.4 Heads and Hands

Our goal in this case study is to annotate videos of people while they are gesturing. This could include a wide range of motions involving any parts of the body, but for this initial attempt, we focus specifically on hand motions. We split

this into two stages: 1) Annotate the locations of the hands in all frames, and 2) determine the pose and 3D configuration of each identified hand image. Figure 2.2 outlines our pipeline. We alternate between human based hand annotations and automatic tracking/clustering. As a preprocess, our videos first need to be cut into shorter sub-intervals, such that each HIT will require approximately the same amount of human effort. All videos remain on our server, and the human annotator communicates with the system through the Mechanical Turk site, which has embedded in it our own Flash based interface.

2.4.1 User Interface in Flash

For the first deployment of the system, we chose to use Flash and ActionScript. Though the system requires a proprietary browser plugin, this plugin is available on all major consumer operating systems and behaves consistently across platforms. The main alternative to Flash is Javascript. For lightweight, HTML-oriented manipulations, this would be the preferred approach. But once the system becomes dependent on subtle, visual feedback– in particular things that require advanced CSS properties in the dynamic HTML paradigm– the likelihood of breaking across platforms goes up. For the sake of rapid, research-oriented experimentation, it is ideal to develop and test on a single platform. In the case of a Flash application, a user will either have the correct plugin or not. And in the latter case, they simply pass over the task and leave it for someone else. This is distinctly different from a traditional website that is intended to work for the maximum number of users and include fallback mechanisms, such as plaintext versions of a page for browsers that do not support a particular web technology.

The skill of motion tracking benefits from experience but is a task that most

anyone should be able to understand and perform with minimal training. The task is potentially tedious in that it involves precise clicking of many points and scrubbing short snippets of video repeatedly. Annotating 10 points in a video of 100 frames potentially requires 1,000 clicks. However, a more efficient formulation is possible. In real video of human motion, particular points may not move at all during a particular range. In this case, by specifying the frame and location when the point stops moving (x_1, y_1, t_1) and the time t_2 when the point begins to move again, we compactly define the location of a point in multiple frames. Another possibility is that a point moves linearly between two locations over a range of frames. Here two (x, y, t) designations can convey many frames worth of annotation. For the sake of decomposing labor, this means it is better for a single individual to annotate a specific point over a range of frames, as opposed to many points for one frame.

A key aspect of the interface is the association of keystrokes with all major actions. This feature greatly speeds up the tracking task since the user can keep his cursor near the action of the video without moving it away to click a button. This is a natural consequence of Fitt's Law [25], which generally says the time it takes a user to click a given point grows logarithmically with the ratio of target distance over target size. In this task, click accuracy and efficiency are of great importance, affecting either the quality of our results or the effective pay necessary on AMT.

The process of tracking a motion is iterative. The user can make a first pass using the forward button to skip 5 frames, create a keyframe, and repeat for a coarse annotation. Then the user plays back the motion to see how the tracking looks. If more keyframes are necessary, as is common in time ranges with rapid or

non-linear motion, the user adds additional keyframes. Our interface also supports zooming, so the user may more precisely locate markers.

2.4.2 Step 1: Tracking HIT

Our first goal is to locate the hands in all frames of the video. An exhaustive approach would be to show the user every individual frame and instruct her to click the center of each hand, advancing to the next frame after each click. But in the case of real human motion, the track of the hands may obey strong spatio-temporal constraints. There are typically spans of time where a hand stays right around a particular spot. In other time spans, a hand will track linearly between two key points. By adding a basic key-framing system to the interface, it is possible for the user to annotate a large number of frames with a relatively small number of clicks.

An early version of the interface had a set of clickable buttons for scanning back and forward along the movie's time dimension. If one attempts the annotation task, it is immediately apparent that having to move the cursor back and forth between these buttons and the main annotation panel is cumbersome. By linking keyboard input to the scan buttons, this problem is solved. The user can quickly skip ahead with the press of a key, while keeping the mouse fixed in its previous location– generally a good starting point for the next annotation click.

If the user can only skip ahead by a single frame, the user may begin to annotate every single frame, like in the original, exhaustive case. Another variation of the interface was designed that provides two sets of buttons and two corresponding sets of keyboard shortcuts. One set of controls skips forward and back by a single frame, while the other skips forward and back by ten frames. The users were instructed

to use the skip buttons as appropriate, but ultimately to annotate with sufficient density to keep the hand within a bounding circle. Many users performed the task perfectly, adding additional keyframes where necessary to capture quick changes in position. Some users took full advantage of the skip-ten feature, and annotated exactly every ten frames, despite the instructions. Many motions are non-linear in nature and if the user samples this with insufficient density, the tracking will be poor.

The simplest approach to improving annotations was to cut the skip interval in half. This tended to slow down power users but caused the average user to put in more keyframes.

In practice, the work performed by crowdsourced labor needs to be validated by a person. For simplicity, this was performed 'in-house' by a researcher, though it could also be performed by AMT in the future. In the case that tracking for a segment was insufficient, the work was rejected and the frames were sent back into the AMT queue for a second attempt at processing.

2.4.3 Step 2: Pose Clustering

After the tracking HIT, we are left with a collection of image patches, all representing the same subject's hands with a consistent camera, scene, and lighting. Given these consistencies, it is feasible to use simple matching techniques, such as normalized cross correlation (NCC). For all patches of a particular hand, we compute all NCC scores. This produces two results. First, by taking the maximum NCC within a window [46] two time-adjacent patches can be more precisely aligned. Second, these maximum scores can be used as rough affinity scores in a clustering approach. (Another option for matching are optical flow based or region based

tracking techniques that perform a full affine warp or more complex warps, but in our experience those techniques frequently fail on complex hand gestures with small image support).

To compute clusters of similar patches, we use affinity propagation [27] on the matrix of NCC scores, taking advantage of Frey and Dueck's publicly available implementation. By setting high confidence thresholds, in a typical video sequence we can reduce a set of patches down to a set of templates that is a fraction of the original number of frames. We can now send these entire clusters to AMT for evaluation in the next stage.

Steps 1 and 2 illustrate a key philosophy of the pipeline: Some tasks are better suited for people, while others are more appropriate for an algorithm. People are far better at accurately locating hands than any currently existing algorithm, but would find the clustering task performed by NCC and affinity propagation to be challenging and tedious.

2.4.4 Step 3: Pose HIT and Configuration Annotation

The next task is to determine the pose of every hand patch. Rather than sending every individual patch for evaluation, we send entire clusters as groups. The clustering is not perfect, so we also ask the user to first validate the cluster before matching the pose. The user can mark individual patches within the cluster as being incorrect, but then proceed to identify the pose for the remaining, correct patches.

One component of the pose-matching task concerns the positions of the fingers within the hand. Considering the raw dimensionality of finger position alone, it is a system of 14 joints, each with 1 or 2 degrees of freedom. It is not surprising that

hand pose has such high dimensionality, given that ours hands' versatility is a key advantage of our species. At the same time, the input videos we are concerned with involve people speaking, using the hands only incidentally. To reduce the complexity of the finger positioning, we developed a set of base poses (as seen in the middle of figure 2.4). By separating finger position from overall hand pose, we can simplify the user interface. The remaining question is how to orient the hand in three-space to align it with the input pose. This can be achieved with three rotations.

The interface provides the user with several base pose buttons that represent finger positioning but not angle. After the user selects a pose, she can tune three knobs to rotate the hand into place. With each adjustment of a rotation knob, the user immediately sees the resulting image and can compare it directly to the input patch. Several options were considered for producing the images for this real-time feedback. One option was to take photos of a human subject's real hand in different, canonical positions. But this would require manual alignment and might not look consistent from frame to frame. Another option was to custom-build a rendering system in Flash to produce graphics. In this case, the time it would take for implementation was prohibitive. We opted instead to use Poser Pro software [36] to render a set of fixed poses. Poser is highly scriptable and using the built-in Python module it was possible to automate the positioning and export of many hand images. We created 7 finger poses, 10 axial rotations, and 7 side-side rotations for a total of 490 images. The third rotational dimension is obtained for free by simply rotating the final 2D image in plane using built-in Flash support including anti-aliasing. By importing the pre-rendered hand images into Flash, we create a Poser "Light" tool that is specifically catered to hand posing. Once again,

the results of this task need to be validated by another person and for simplicity this was done by a researcher, but could be handled by AMT in the future.

2.4.5 Step 4: Post Processing Poses

After step 3, it is possible that many hand patches have not been estimated. If a hand was placed in the wrong cluster and identified as such by a user in the cluster validation phase, we do not know its pose. One approach would be to send any rejected patches back to AMT for a second attempt. But this becomes expensive because now a full HIT must be devoted to a single patch. Our approach is to backfill the data using NCC-based post-processing.

For any unidentified patch, we compute the NCC distance to the image patch of each pose cluster center. We then can compute the weighted average of the closest K clusters using the NCC values and pose annotations of that specific cluster. (To compute the weighted average, we need to convert the pose angles into the twist representation for better interpolation properties [53]). Optionally we can do this on existing pose annotations for further fine tuning. Based on a preliminary sampling of 200 random frames, this technique had an accuracy of 77% correct pose estimation. We are currently using these predicted values in some experiments (see below), but do not use them as "ground-truth" training data.

2.5 Case Study

We previously collected a database of 189 videos of politicians and public figures giving speeches. Our efforts concerned processing all videos with automatic motion estimation and machine learning techniques (similar to bag-of-feature rep-

resentations that do not need explicit labels, only coarse motion-style targets) and we got promising results for various recognition and clustering tasks [97, 98].

For the next data collection phase, we intend to fully annotate this database with hand locations and pose. Our AMT based system produced $23,667$ labelled video frames on a smaller pilot database of 10 videos. All our analyses in this case study are based on this small pilot database.

This pilot database was labeled with 158 "Tracking HITs" containing $6,769$ keyframe click annotations and 1612 "Posing HIT" annotations. We experimented with different pay rates for the two types of HITs, and got good results with \$0.25 for each 150 frame Tracking HIT and \$0.05 for each Posing HIT that contained on average 9.5 patches. The total cost was around \$120. If we had used a single frame approach, asking users to label the hand position and pose for each individual frame, this would require as many HITs as input frames. Assuming this task costs at least as much as a single Posing HIT, the total cost is more than \$1000 and annotating the full video database would cost over $\$10,000$.

Both the low price of HITs and the good quality of the results surprised us. We had a savings of an order of magnitude over a frame-based AMT approach or compared to using a professional service[1].

We have begun to use this data for two applications: for interesting visualization applications by domain experts, and to increase the annotation richness of training data for our automatic gesture tracking and classification system.

For assessment purposes, we built a preliminary suite of visualization tools that can reveal frame-based, gesture-based, and subject-based aspects of the pilot

[1]If we contracted with a professional studio, it would have taken each tracking artist between $4-20$ seconds for the tracking task of two hands per frame and $8-40$ seconds for the posing task [2]. At a rate of $\$50/h$, the pilot database would cost $\$6,000-\$30,000$.

database. Figure 2.5 shows visualizations of the different "kinespheres" for different subjects. Figure 2.6 shows a few automatic matching examples of typical gestures in one example video to a specific subject or to the entire database. This allows us to analyze body-language in video more quantitatively, for example in counting the occurrence of specific gestures, poses, and other attributes in a speech.

Feature extraction and dimensionality reduction Dimensionality reduction is often used for visualization of high-dimensional datasets. In our case, we use such a visualization to observe structure amongst and within gesturing subjects. Two primary aims are to 1) note similarities and differences amongst subjects and 2) discover atomic "movemes" characteristic to, and shared amongst individuals. Both tasks can be performed by watching the (annotated) videos, but this is extremely time-intensive.

For each video frame, we first convert the output of the tracking and pose annotations into a 24-dimensional vector representing velocity, pose orientation (twist), and pose type (1-of-K encoding where $K = 7$) for each hand. While we could directly apply dimensionality reduction to this representation, we first perform a type of feature extraction that extracts an overcomplete, latent binary representation from the original data [84]. The reasoning for this step is twofold. First, our feature extraction method explicitly captures the dynamics of the gesture data. Each binary latent vector actually represents a local window in time. Secondly, the latent representation is *distributed*, where many different latent variables can interact to explain the data. This abstract representation is a more salient input to the dimensionality reduction step. For example, we do not have to worry about weighting velocity vs. pose type or orientation: this has already been figured out by the dynamical model. Nor do we need to worry about how the heterogeneous

input types (i.e. the velocity and twists are real-valued but the pose type is multinomial) affect dimensionality reduction. In our experiments, we use 100 binary variables and a third-order dynamical model. Therefore the output of the feature extraction stage are length-100 binary vectors for each (overlapping) window of 4 frames. To this representation, we then applied t-SNE [91] to reduce the vectors to dimension 2 (Figure 2.7). We note that while person labels have been used in preparing this plot (to add color), both feature extraction and dimensionality reduction are completely unsupervised.

2.6 Deployment at the Snowbird workshop

We deployed our system as part of an effort to analyze the nonverbal communication of academics. We digitally recorded all of the contributing and invited speakers at the Learning Workshop, held in Snowbird, Utah between April 6-9, 2010. We recorded 28 speakers, with talks ranging from 10-40 minutes each. After each block of speakers, video excerpts were chosen, then sent into the first section of the pipeline described in this chapter. (We did not run the pose estimation stage.)

We were able to obtain accurate hand and head tracks for each of the speakers within hours of their talks. This allowed us to perform a simple analysis of the academics which included 1) clustering motion tracks; 2) ranking speakers based on the "energy" exhumed in each talk; and 3) locating specific gestures (e.g. handwaving) across speakers. We presented our analysis during the concluding session of the workshop, to demonstrate the automation and quick turn-around of the system. This experiment demonstrates the flexibility and efficiency with which

data can be collected for "on-site" gesture analysis using crowdsourced motion tracking.

2.7 Extending the Marker Set

The second version of our interface, shown in 2.10, was built in Javascript and takes advantage of HTML5 for playing video files. A user is typically asked to track 3-4 points over 100-200 frames using it. Each part appears as a clickable colored square as in a paint program. When the user clicks on the video panel, a marker of the selected track appears. A line is also made underneath on the appropriate track visualization to indicate a keyframe has been placed at the current time index. This helps to remind the user of the keyframe process. Some users get into trouble when they place a frame incorrectly, then scrub back near the frame and try to fix it by adding additional keyframes. If the original incorrect keyframe isn't removed or fixed, the motion will always have a problem. Users can change the position of previously placed markers simply by dragging. Users can jump to a particular track and keyframe by clicking the lines on a track visualization.

We indicate the points to track with 4 letter symbols, like HEAD, LANK, RELB, which correspond to typical marker sets used in motion capture. We include a diagram with each part marked on a person and display a textual description to go along with the selected track. An important detail with motion tracking is to prevent the user from annotating with a left-right flip. If the video contains a subject facing out, then body parts are mirrored relative to the user. We can define left to be the user's left or the subject's left, but either way some workers may get it wrong. In our heads and hands interface, we devised the simple solution of popping

up a large warning that appears anytime the user seems to have it backwards. For baseball annotations, we describe the LHND point as "the hand wearing the glove" and the RHND point as "the throwing hand," and this effectively prevented flipping. (Note that we were only studying right-handed pitchers.)

2.7.1 A Qualification Task

In our first attempts at markerless motion capture on AMT, we simply checked all completed assignments and determined which to approve or reject. This took up a large amount of researcher time and would not scale. Next we attempted to deploy HITs redundantly with multiple assignments and use the median to improve accuracy. This required more total work for AMT and we still were not satisfied with the general quality of results. Many workers would annotate just a single frame and submit the assignment while others clicked randomly or otherwise achieved very low accuracy. In the end, we still had to manually check most of the work.

We now require all workers who wish to complete our motion capture HITs to pass a qualification test. We use the exact task that is demonstrated in a video that plays during HIT preview. The user must track 3 points over 90 frames of a baseball pitcher throwing a ball. Our test is scored automatically using as an answer key, a completed annotation by a tracking expert in our research group. A user attempting the qualification test is allowed to request a score at any time, which is computed automatically by comparing all 270 data points to our answer key. The result is a sum distance which is then scaled to a score out of 100%. We scale the test such that a score of 80% is satisfactory but not perfect. A user attempting the qualification test is only allowed to submit their work after they

Table 2.1: HIT Details. Each marker is annotated by one worker.

Frames	Tracks	HITs	$/HIT	$/pitch	$/min
100	13	4	$0.50	$2	$36

have obtained 80% or higher and we record the final score. By allowing some range of quality, we have created a kind of honeypot to mark workers who may be less motivated. For some HITs, we can accept our minimum 80% quality score while for others, we want the most talented and motivated workers. One recommendation we have is to pay users to take qualification tests instead of making them unpaid, like most on AMT. This helps to increase the labor pool and with the auto-scoring system, you only pay for satisfactory completions.

2.7.2 Cost

Using the previously described interface and qualification system, we are able to annotate complex footage of human motion at a price considerably lower than what a studio pays for comparable tracking services. We typically create HITs that involve annotating 3 tracks over 100 frames which takes 5 to 10 minutes when done efficiently. We pay 50 cents per HIT, resulting in an hourly wage of $3-$6. Table 2.1 shows our effective cost to annotate pitches. This is the cost to get 13 markers of one pitch tracked by one worker each. We typically deploy each HIT 5 times and use the median of each marker location to boost the quality of our tracking. Of course, this costs 5 times as much.

2.7.3 Accuracy

To assess the accuracy of our human-powered markerless motion capture system, we used traditional motion capture data as ground truth. We recorded a baseball pitcher in our motion capture studio along with video data, then performed camera calibration so we could project the 3D data to align with the video in each frame. We divided the tracking task across 4 HITs, each one 115 frames with 3-4 tracks. We deployed each HIT to approximately 30 AMT workers.

We denote the true location of a particular marker:

$$P = \begin{bmatrix} x \\ \\ y \end{bmatrix}$$

The location of a marker at time t is P_t. We define $\hat{P}_t$ as an estimate of a marker's position and compute it by taking the median of any number of user estimates, as obtained from our annotation tool. We define pixel error for any marker at a time frame as the L2 norm of the difference between the estimate and ground truth:

$$E_t^{Pixel} = |P_t - \hat{P}_t|$$

Thus pixel error for a given marker is simply the per frame error averaged over all frames. Table 2.2 shows a summary of the results. All numbers are reported in pixel values and the input video was 640x480. For reference, 10 pixels is approximately 5 cm in the real world given our particular camera configuration. There is a considerable amount of error and it varies across the marker set. The knee and ankle seem to be the easiest to track and this could be because the lower markers

move slower than the upper ones in a baseball pitch and also because this area of the video had higher contrast. In the case of the hand and elbow markers, there were many frames where it was virtually impossible to distinguish the pitcher's hands from his torso. This is a limitation of our recorded video. Further examination of the errors over time showed that workers had trouble consistently locating subtle 3D configurations in 2D. For example, they were asked to mark the head as the point on the surface of the bridge of the nose between the eyes, but the majority of workers clicked the center of a circle that approximately bounds the head. This means they were actually tracking a point in the center of the skull. As an alternative assessment, we propose a measure of the motion itself, denoted in the table as Motion Error. We define the motion for a marker at time t as the vector:

$$M_t = P_t - P_{t-1}$$

We compute motion error for a particular frame and marker as:

$$E_t^{Motion} = |M_t - \hat{M}_t|$$

Again, we average across all frames and report the errors in the last two columns of Table 2.2. Using this measure, a worker may have the wrong conception of a marker location, but if he tracks the wrong location correctly, motion error should not be affected. Note that pixel error and motion error cannot be directly compared since motion error is a velocity measured in pixels/frame. We can still look at the relative errors between different body parts to see, for example, that workers get a similar amount of absolute pixel error when tracking shoulders and hips, as compared to hands and elbows. But when we look at the motion error, we see

Table 2.2: 2D Accuracy of Motion Capture

	Pixel Error		Motion Error	
Part	L	R	L	R
Shoulder	9.92	10.15	1.96	1.89
Elbow	14.97	11.02	2.87	4.03
Hand	10.35	11.28	3.26	4.26
Hip	10.57	11.18	1.43	1.56
Knee	8.54	5.37	1.82	1.30
Ankle	7.14	6.42	1.26	1.31
Head	8.93		1.24	

that the workers do quite a bit better with tracking the motion of shoulder and hips. This makes sense because the shoulders and hips move slower than elbows and hands.

Despite the reported error, the animations of markerless motion capture seem to capture some large gestalt of a subject's motion. The workers are relying on their own perceptual faculties to perform tracking, which may not be ideal for absolute tracking. A person viewing the resulting animation has a perceptual system comparable to that of the original annotating worker, which could explain why, subjectively, visualization results look good.

2.7.4 Crowd Effects

We are hoping to take advantage of the power of the median to provide the most robust estimates for our tracking data. The Wisdom of Crowds [82] describes many historical examples of crowd success and a few failures while [35] studies this explicitly with a guessing game and reveals that crowds are not immune to systematic bias. To study this effect as it relates to motion tracking, we deployed certain baseball annotation HITs to many workers and studied the change in error as we

included larger sets of workers in the crowd. We selected random subsets of the 27 workers, ranging in size from 1 to 27. For each subset, we take the median tracking result and compute the error of every point relative to ground truth, as described in the Accuracy section. We repeat many times with different permutations to reduce noise. We take the mean of all of these errors and report the results in Figure 2.11. We note that error decreases for all markers as we increase the size of the crowd but that most of this averaging advantage is realized with 5 workers. Applying a linear regression to each marker line in the range from 10 to 27 results in slopes around -.01. This means, assuming the linear relationship holds, it will take 100 additional workers for each desired additional pixel of accuracy. This appears to be prohibitively expensive, so more research needs to be done to improve tracking tools or the instructional design of the HIT.

2.7.5 Visualization and Search

A result of our effort is the ability to visualize motions. One possible visualization is video playback of the annotation as a stick figure. We can compare stick figures side by side for an analysis that focuses on the motion instead of pixel-level details. We can overlay a trace of any marker over time to emphasize and convey the shape of that marker's motion. (See our supplemental video for examples of this technique, located at http://movement.nyu.edu/markerless.) The second option is to create static images that summarize a motion by concatenating several frames along with marker traces as in figure 2.12. With this technique we can illustrate a complex motion with a single, static image.

Another result of our system is the ability to search for gestures by example. We used the AMT-based annotation pipeline on a set of academic videos from a recent conference at Snowbird in Utah. Similar to a search engine on text, or the new Search by Image feature on Google, we can build a fast search engine that is motion or gesture based. We store all videos that are annotated with our AMT pipeline in a database and index the database entries with a motion-based hashing technique that uses the head and hand motion vectors in a window of frames. The hashing scheme is designed to be robust to changes in global position or scale variations. Figure 2.6 shows an example of inputing a video with a hand-waving gestures and we get back a ranked list of videos that have similar motions.

Another application of this crowd-facilitated gesture annotation can be used for body language analysis of speakers. In previous research [97, 98] we developed an analytical technique that can summarize what most frequent gestures for a specific speaker, or how two speakers compare to each other, i.e. how many idiosyncratic gestures they have in common.

2.7.6 3D Reconstruction

Until this point, we've only discussed our system as a way to recover 2D motions. In general, recovering 3D from a single 2D image is ill-posed, especially in cases with complex, non-rigid motions. However, when multiple camera views of the same scene are available, it may be possible. We took archival footage of Mariano Rivera, a well-known baseball player with a distinct pitching style, that showed him from several different angles and ran each view individually through the pipeline. We applied a factorization-based 3D reconstruction technique (related to [89]) that can reconstruct the body markers with high accuracy in 3D.

This was then used to transfer to a 3D animation package and used for some baseball visualizations for a project with the New York Times [64]. The system produces skeleton motion of the baseball pitcher including subtle, high-speed dynamics. With some artist cleanup, the final result is a compelling 3D animated re-creation of a human motion that was never seen in a motion capture studio.

2.8 Discussion

The system here described is a proof of concept for optimizing large, video-based annotation tasks. Our pilot study shows the potential multiplier effect obtained by combining human intelligence and artificial intelligence in a single pipeline. By proceeding with experiments on our entire motion database, we hope to show that we can get even better than a linear speedup. If an input video is large enough, we believe performing hand tracking on a small part of the input, will provide enough training data to build an automatic detector for filling in other frames. Likewise, as input size increases, pose clusters will grow, and it will take less human effort per pose on average.

For the next iteration, we envision a more generalized pipeline in which certain tasks can be attempted interchangeably by an artificial or a human intelligence. For example, an AI hand detector could attempt to locate the hands in a video frame, and only if the confidence value is low, does that particular task need to be sent to a person. Both HIT and AI results could go to the same approval queue. As the model's training and performance increases over time, human effort can be reallocated toward approval tasks, away from the more time-intensive tracking and pose identification.

This work demonstrates the possibility of a markerless motion capture pipeline powered by human intelligence. With a keyframe-based user interface, we make a task that can be performed cheaply by a general pool of information workers with a resulting signal that can be boosted through redundant task deployment. We note that our system is limited in terms of absolute pixel accuracy but nonetheless, creates nuanced visualizations of motion. This can be observed in the various animations we have produced of natural human gesture, stylized walks, and sport motions. Subjectively, these animations capture some essence of a subject's motion including subtle dynamics like bounce. Our use of a complex, auto-scored qualification test allows us to simultaneously train workers and assess their ability. We believe that a harder qualification test requiring more exact positioning of markers could result in a better labor pool and better absolute pixel accuracy for future work. We also plan to develop an interface for workers to smooth jittery motions by applying bezier-based smoothing and manual adjustment of control points.

We show several applications of our 2D annotations including static visualization and animated playback. We show that 2D tracking of just 3 markers is enough to facilitate example-based search across a corpus of data or to create single-subject motion profiles by clustering. Finally, we demonstrate that several 2D annotations of one subject from different camera angles can be used to create a viable 3D reconstruction even with a motion as nuanced as a baseball pitch.

The pipeline is invaluable for our work in human motion analysis. It opens up the possibility of capturing motion from historical video archives or for future recordings in scenarios that are incompatible with a traditional motion capture studio.

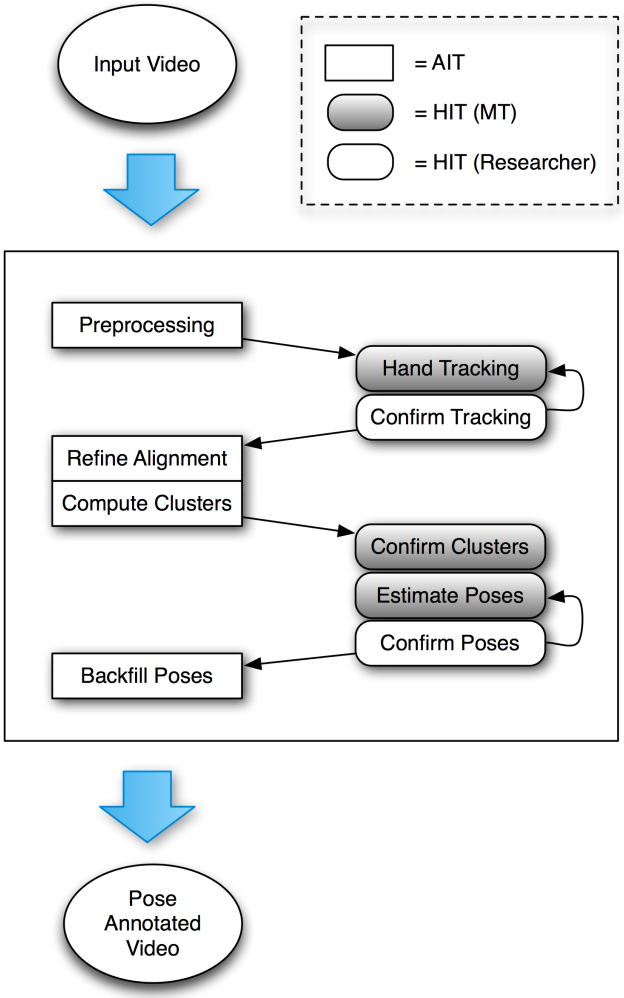

Figure 2.2: Information flow between artificial intelligence tasks and human intelligence tasks.

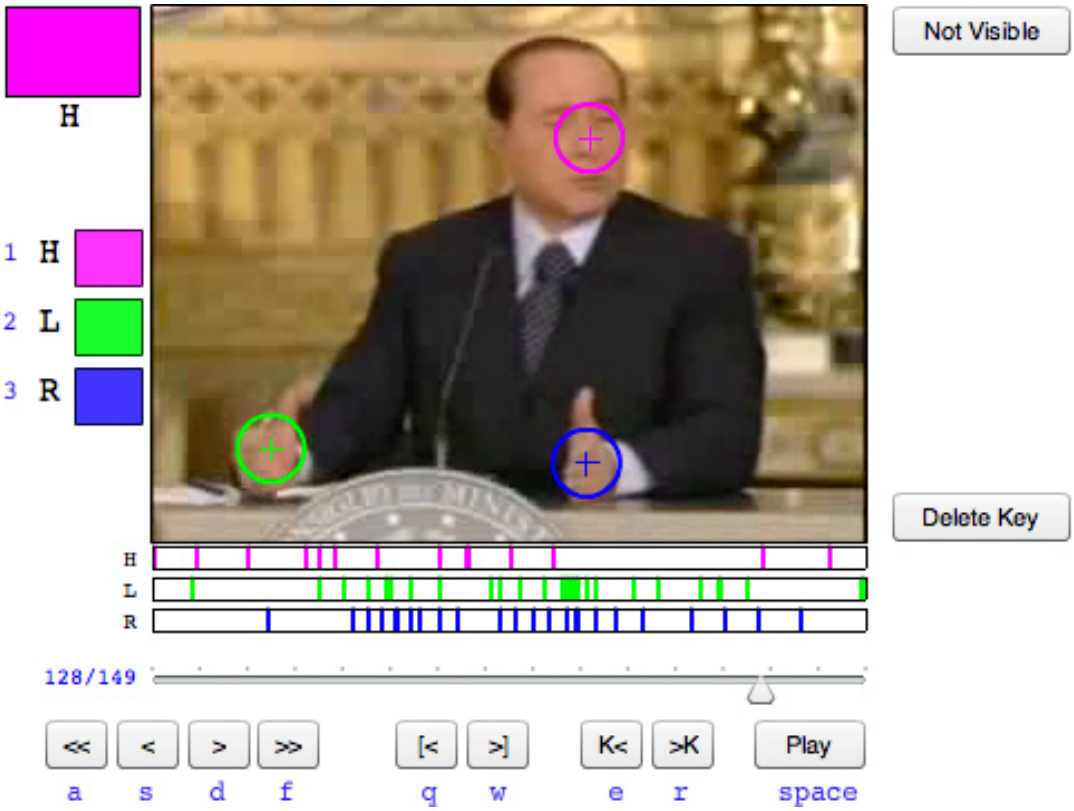

Figure 2.3: Our first tracking interface for Mechanical Turk.

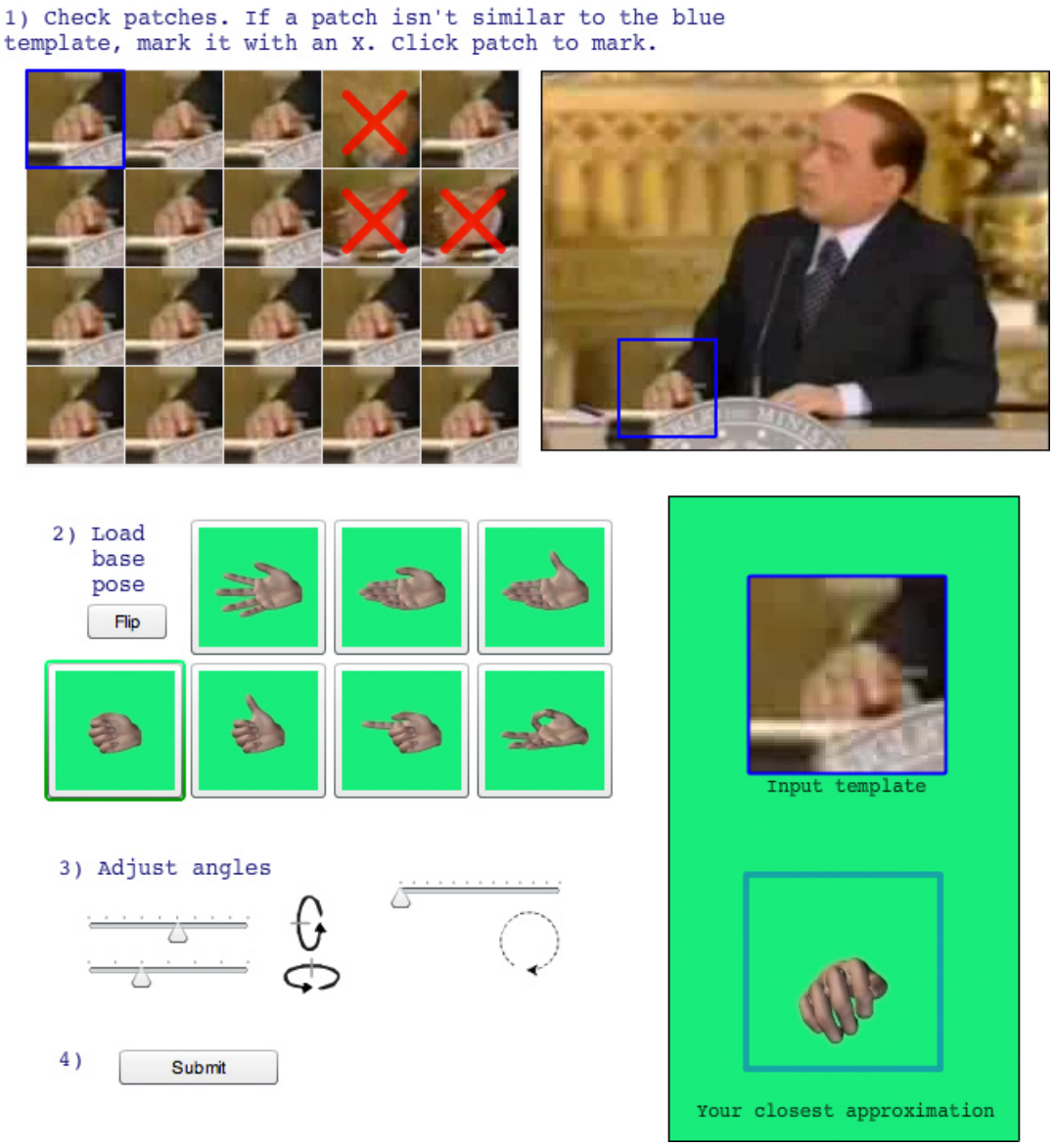

Figure 2.4: Our cluster verification and pose estimation interface for Mechanical Turk.

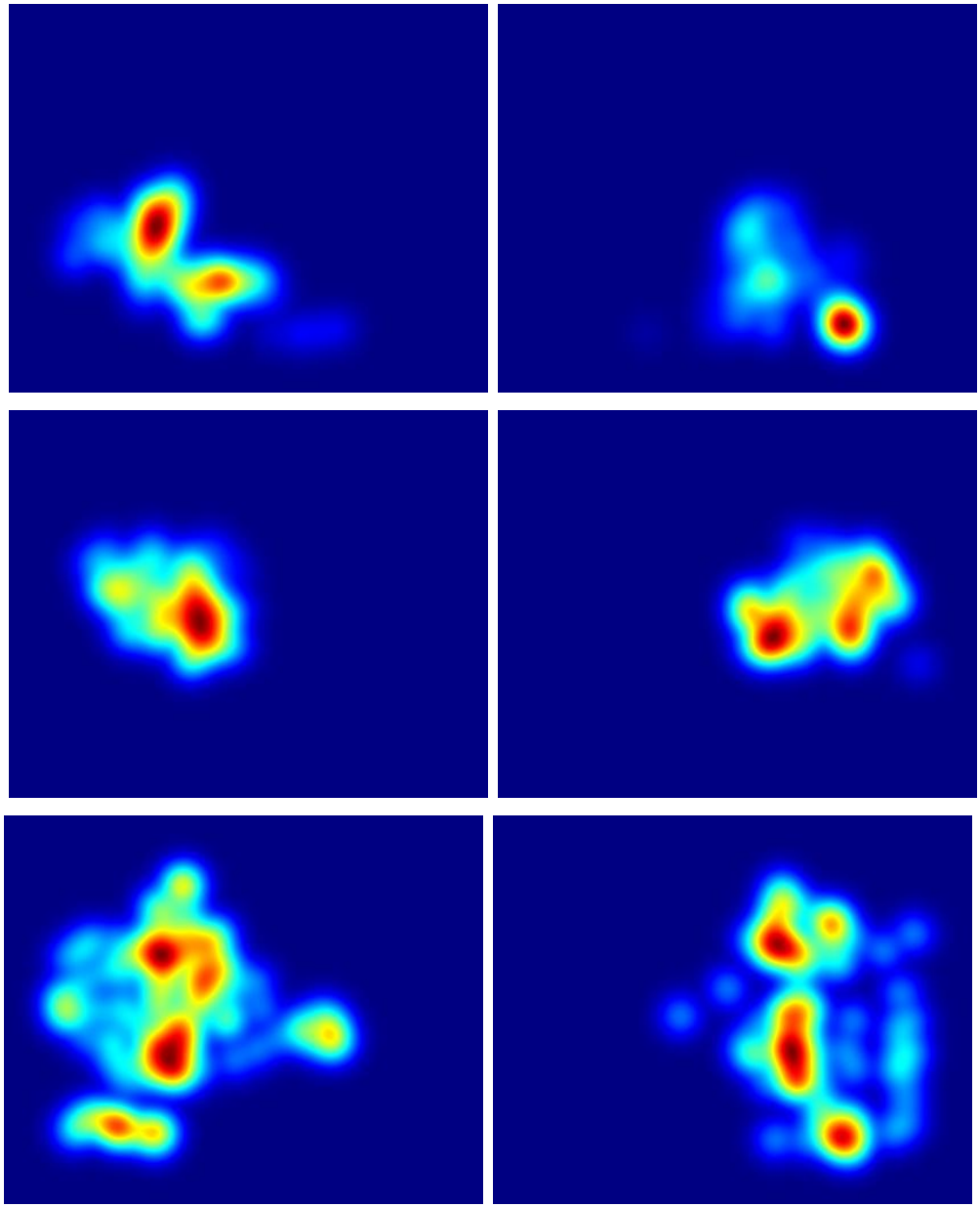

Figure 2.5: The hand gesture kinesphere for different subjects. (Video is not mirrored, so left hand kinesphere appears on the right and vice versa.) Barack Obama (first row) has a stronger bimodal distribution with his right hand. Hillary Clinton (second row) shows this effect more on her left. Vladimir Zhirinovsky (last row) has a wide distribution of gestures and the largest kinesphere.

Figure 2.6: Example gesture matches within subjects and across subjects. Each column has similar gestures. Any window of motion can be compared to all other windows of the same length using an L2-distance. By applying non-maximal suppression, we find good candidate matches, that can then be verified visually.

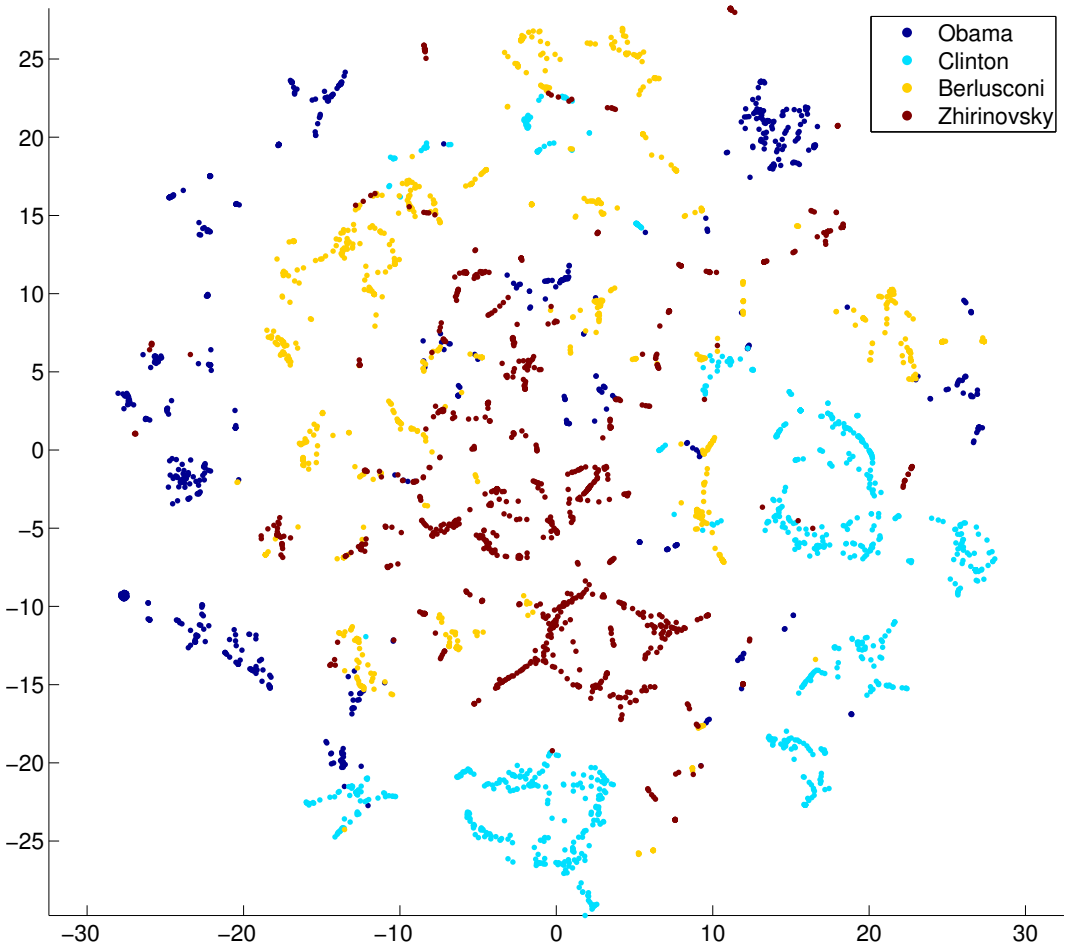

Figure 2.7: Dimensionality-reduced annotations. Data points represent overlapping 4-frame windows. We note that frames of a particular subject tend to cluster together. At a finer grain, we note the emergence of "strokes" in 2D which correspond to spatially and temporally local "movemes" in video.

Figure 2.8: An example search of the Snowbird dataset for a hand-waving gesture.

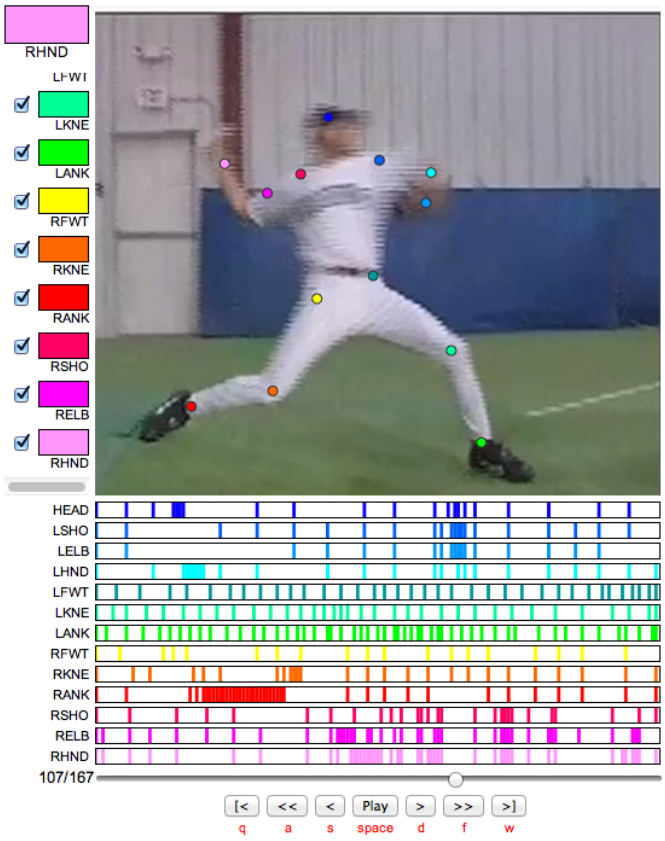

Figure 2.9: Our user interface for tracking, showing a completed 13-point baseball pitch annotation. Each marker and its corresponding track is indicated with a different color. Each vertical line represents a defined keyframe.

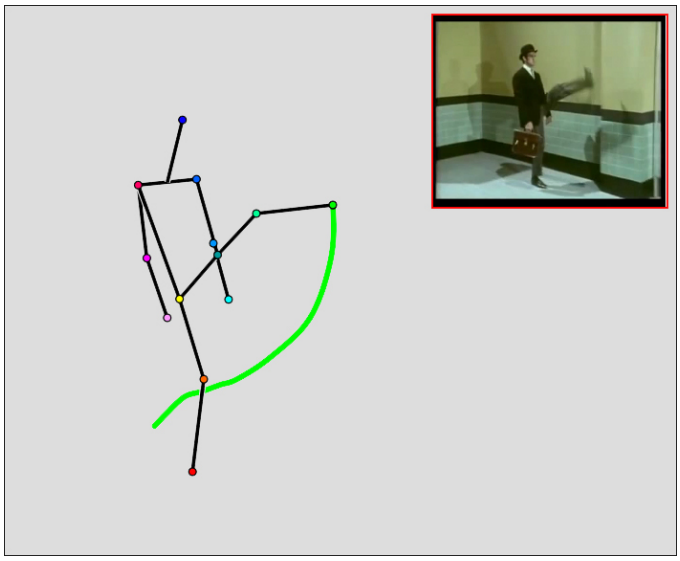

Figure 2.10: Motion visualization. Here we see a reconstruction of John Cleese's iconic Silly Walk.

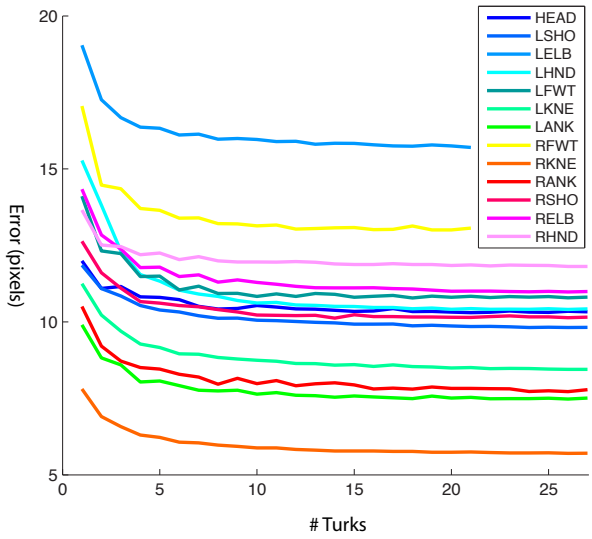

Figure 2.11: The median error for tracking 13 points, measured in pixels, shown for different crowd sizes.

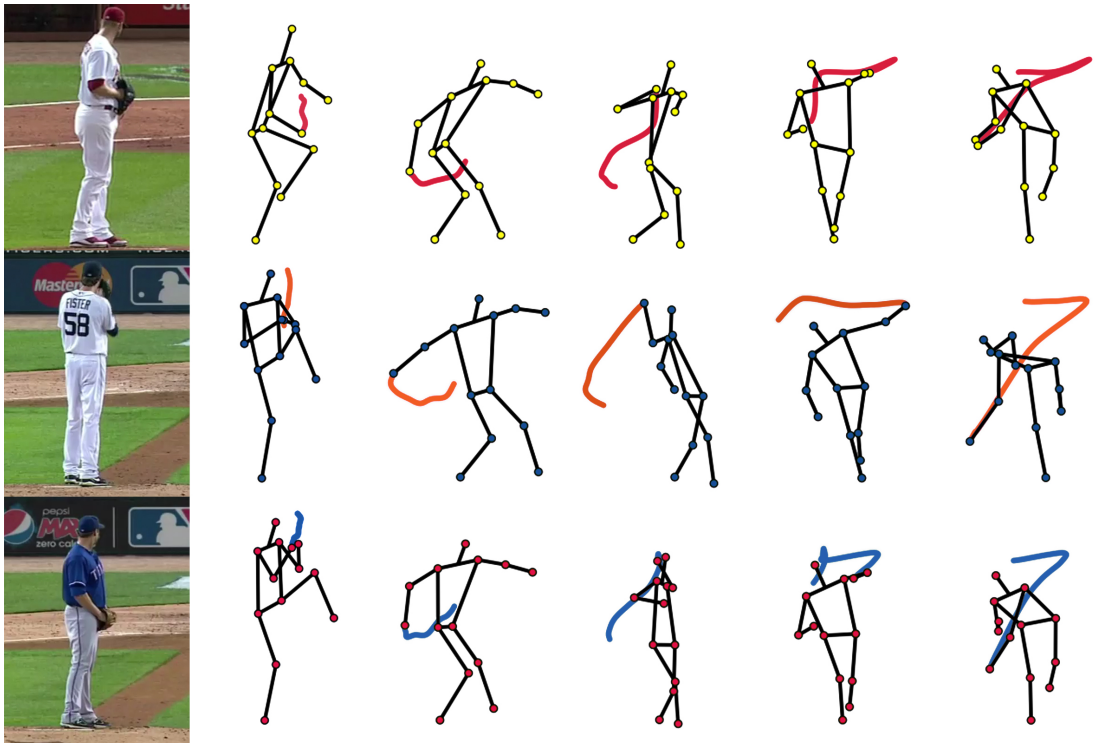

Figure 2.12: Example motion summaries from our system. Each row is a different MLB pitcher (Chris Carpenter, Doug Fister, and Colby Lewis), shown at five different stages of a pitch (knee up, arm back, foot down, ball release, and follow through). The motion trail of the right hand is shown for frame t-10 through t. While all three deliveries follow the same general pattern, each pitcher has a unique signature to his motion. In the knee up stage, Fister and Lewis raise their left knees almost to the level of their left hands but Carpenter's left knee never goes above his waist. Leading up to ball release, Fister moves his right hand along a flat plane, while Carpenter and Lewis follow a diagonal path. We also see that Lewis twists his right arm in between the foot down and ball release stages. In the final stage, Carpenter's follow through is quite sharp, with his arm moving back along the same path as his ball release, whereas Fister and Lewis follow v-shapes during follow through.

Chapter 3

Privacy Enhancing Video Filters through Crowdsourcing

3.1 Introduction

The rise of video technology has raised concerns about privacy for many years. Real-time video communication has been suggested for a variety of contexts and has been demonstrated to greatly augment audio channels with non-verbal queues and body language. But the flip-side of this increased intimacy is the potential for embarrassing personal details to appear on screen or in archived footage. Social scientists have explored attitudes toward sharing and privacy while computer scientists have attempted to offer technical solutions. The hope is to develop video technologies that can avoid or mitigate privacy concerns but still offer utility in certain collaborative contexts.

Earlier work in this domain focused on practical, work-related scenarios, as digital video cameras and high-speed internet connections were expensive. But

the modern age of social media combined with the ubiquity of smart phones and web-cameras make video sharing more prevalent than ever. In the shift toward social computing, the potential for embarassing breaches of privacy is even greater. The popularity of Chatroulette [86] shows that many people desire to communicate through video on the Internet with complete strangers. In parallel, the success of the mobile application Snapchat [1], which destroys shared images shortly after viewing, demonstrates that many will choose privacy over persistence.

Our research lab focuses primarily on the study of human motion, either in the context of traditional, studio-based motion capture or quantitative analysis of video of humans in motion. For much of our work, we don't want to know the identity of our subjects, yet this information is implicitly bundled in video. Thus we seek methods for obfuscating video to hide identities while retaining useful motion signal. Through our work, we have gained a great appreciation for the expressiveness of the human form and learned that a surprising amount of expression comes through in a small set of animated dots, as in the case of motion capture data. This suggests to us that motion-centric filtering techniques have great potential for obscuring human identity while still allowing useful body language to come through. Such a filter would aid in our research compliance and could also benefit users of video-based social media where privacy concerns abound.

3.2 Related Work

Much work has explored the possibility of obfuscating video to hide certain details while still allowing useful signal to come through. The earliest attempts used

filtering techniques that could be described as linear: mathematical operations that could be implemented with simple matrix operations and, even a decade ago, ran in real-time. In subsequent years, most work in the area of video privacy has taken a more sophisticated approach. For example, face-detecting neural networks can be used on video frames to systematically hide faces. This is made possible with today's increased computing power and advances in machine learning. Mobile phones and digital cameras can now detect faces in real-time and do so with accuracy in the 90% range. It's important to note, however, that no such detection method is 100% accurate and probably never will be. In the case of video privacy, even a single false negative could reveal a person's identity. It is our opinion that the early, linear approaches are worth revisiting in the modern era of ubiquitous video. Using crowdsourcing as an engine for performing quantitative, perceptual experiments, we can assess a variety of filters.

Many papers have attempted to address the challenge of video privacy through image processing. [68] contains a useful survey of 12 such works and divides them into two basic categories. *Selective obfuscation* uses some kind of detector to find bodies or faces while *global obfuscation* techniques generally process all pixels with global operations that modify the entire image.

3.2.1 Selective Obfuscation

These techniques rely on face detection as a top-level signal for privacy preservation. Depending on access policies, certain viewers will only be allowed to see video with faces blurred or completely blacked out. Such approaches are all vulnerable to a fundamental limitation of object detection. There is a low but non-zero probability of a false negative. One false negative in a series of video frames may

reveal a person's identity. A conservative approach to privacy cannot accept the non-zero risk of a false negative. So for our research, we focus exclusively on global obfuscation.

3.2.2 Global Obfuscation

Comparatively little work has explored the use of global filtering techniques. In [8], researchers conducted experiments with pixelation and blur filters. Twenty test subjects were shown videos of actors in office scenarios and asked questions about the number of actors, gender of actors, and their respective activities. The independent variables were filter type and strength, which was either blur radius or pixelation block size. The general result, as one might expect, was that test accuracy correlated negatively with filter strength. To assess the privacy-preserving ability of the filters, test subjects were asked to imagine themselves in each video scenario and rate on a numerical scale how well their privacy would be preserved. This work is a useful starting point but is limited by small sample size and the challenge of resolving the quantitative test results with subjective privacy assessments.

The work in [54] focuses on using a blur filter to support privacy for telecommuting. The telecommuting scenario creates an environment where someone may perform both work and personal activities such that an *always on* video system can both aid productivity and harm privacy. Researchers created videos in which actors appeared in a number of scenarios ranging from mundane work tasks to embarassing grooming activities to full nudity. Using a similar approach to [8], it was determined that no such level of blur existed that could offer useful work-oriented context and still preserve privacy in the case of the more extreme scenarios.

In [101], a single pixelation filter was considered along with several other filters. They also tried edge-detection and shadow image filters. Shadow filters were essentially frame differencing to create a ghost image that conveys an approximate form of an actor's outline. Twenty test subjects tried to identify work activities for each filter then tried identifying the actors in various scenes. All filters except the shadow view resulted in activity recognition in the range of 90% to 100% while the shadow view had 60% accuracy. For identity tasks, the shadow view resulted in 25-30% accuracy while the others ranged from 90% to 55% accuracy.

3.3 Methodology

Given the complexity of privacy and how it might pertain to all the possible informational flows in video, we choose instead to focus on a quantitative measure of *identifiability*. Our stated goal is to assess the potential for video processing to preserve privacy while still revealing body language. We use the ability of our test subjects to identify a particular human as a proxy for privacy. We use their ability to correctly identify gestures in videos as a measure of body language transmission.

3.4 The Study

We developed an experimental framework for exploring the perceptual effects of various video treatments. The basic idea is to show test subjects processed video of human gestures then quiz the subjects to see if and when they can identity the people or their body language. The key challenge in developing the experiment was in creating the underlying dataset. The work in chapter 4 generates corpora of human gesture. It does so in the context of a game with a purpose, first proposed

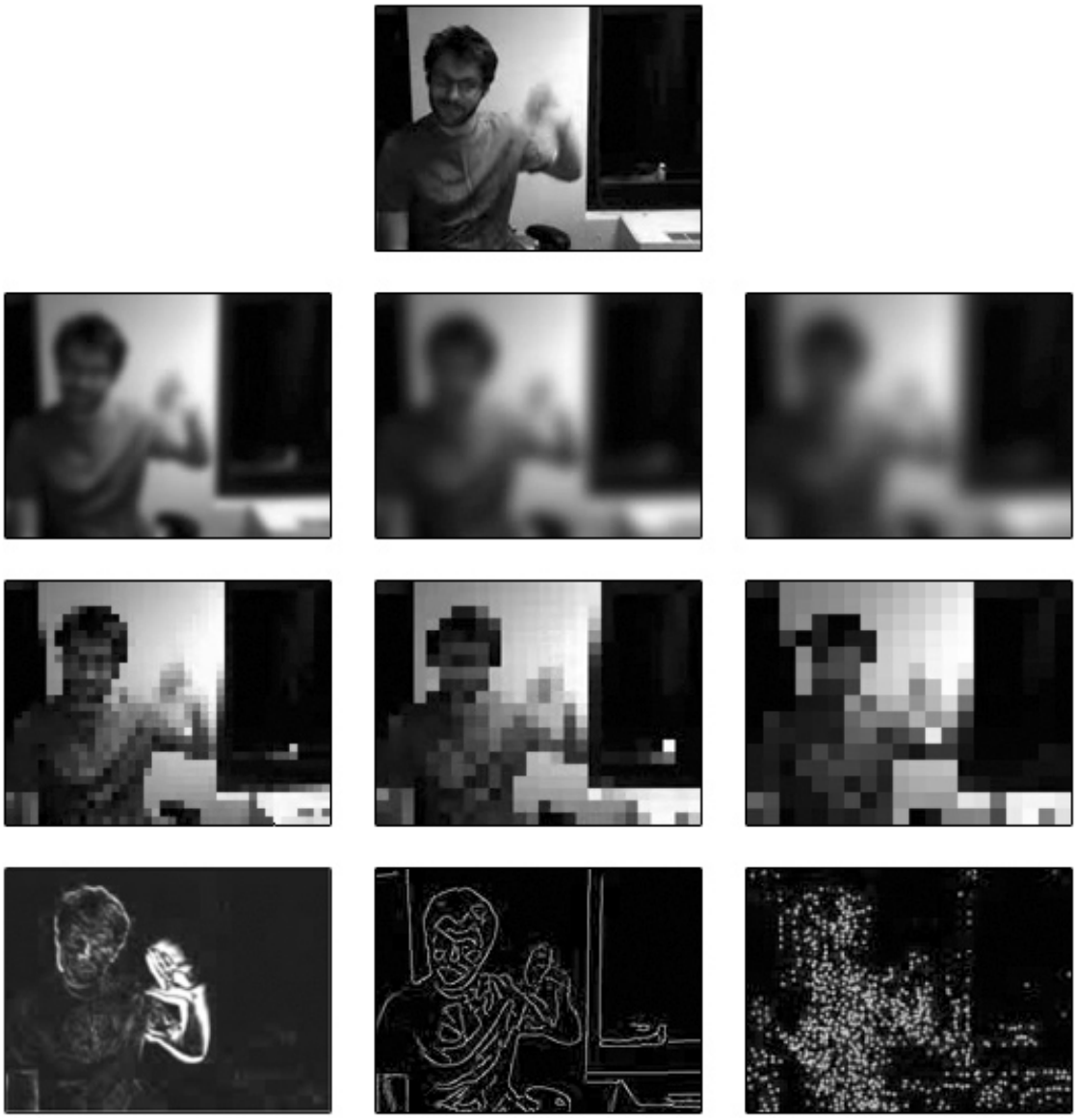

Figure 3.1: The candidate filters used in our experiments. Top row: original image. 2nd row: Gaussian blur at 3 different strengths. 3rd row: Pixelation with 3 different block sizes. 4th row: frame differencing, Canny edge, and greendot.

in [92]. We use such an approach to crowdsource the collection of a dataset that is specifically catered to our experimental needs. Users were prompted to either invent gestural responses to text prompts or to copy the gestures in sample videos. Through this method we obtained data that was nicely constrained. All video data

was generated on desktop computers with stationary cameras at VGA resolution, but actors, lighting and scene across videos varied considerably. This stands in contrast to many academic datasets, where graduate students act in parking lots or conference rooms [15]. The nature of our data collecting game took place asynchronously over many days so any particular user may appear with a variety of outfits and with different backgrounds. This variation is essential so that test subjects are forced to make identity determinations based not on background images or clothing but on the appearance of a person's body and face.

We show test subjects processed videos of human gestures then quiz the subjects to see if they can identity the people or their body language. Our results show that motion-centric filters, such as frame-differencing and greendot offer the potential to obscure identity well while minimally impacting the ability to identify certain gestures.

3.4.1 Gesture Corpus

Over the course of several weeks, this game-based approach was used to collect gestures from approximately 50 volunteers who created over 400 short video clips.

From this corpus, we had enough data to drive two categories of experiment.

3.4.2 Identity Experiment

We show the test subject a video of a target person making a gesture. Underneath the target person, we show a lineup of clips of people all performing the same gesture, which is different from the target gesture. One member of this lineup is the same person shown above. We also ensure that the two videos of the target person have different backgrounds and the person is wearing different clothes.

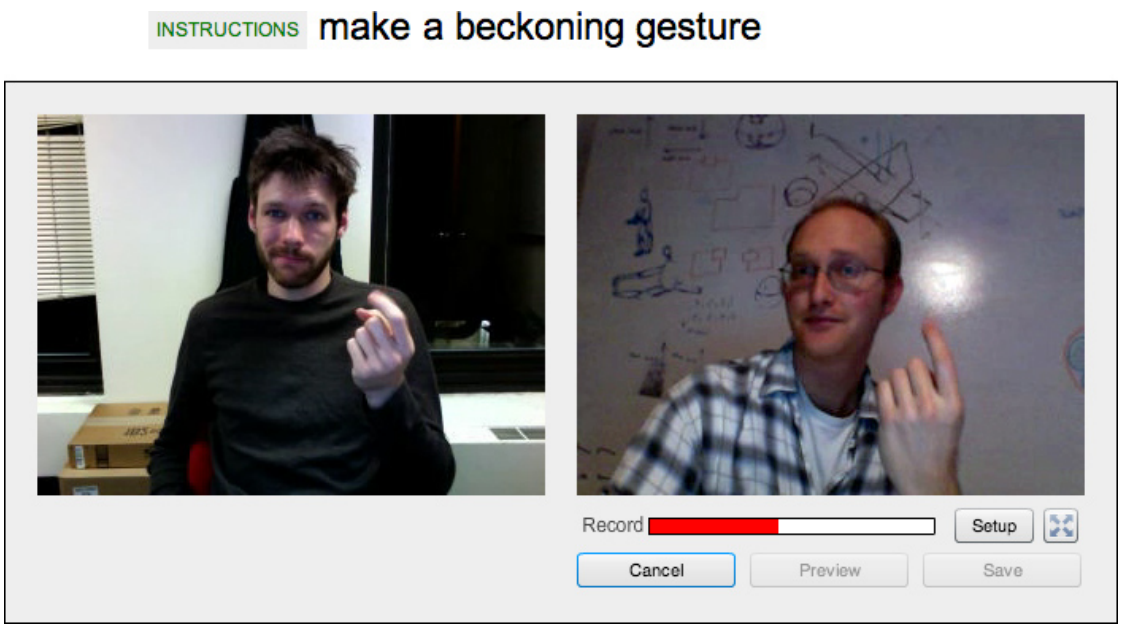

Figure 3.2: A game with a purpose that encourages users to build a corpus of gesture data.

Figure 3.3: The ten gesture categories from table 3.1 with arrows to indicate motion.

3.4.3 Gesture Experiment

We show the test subject a video of a target gesture. Underneath the target gesture, we show a lineup of clips of people all performing different gestures, with a single clip of the matching gesture shown above. The person performing the target gesture above does not appear in the lineup.

Gesture	Meaning	Count
Nod yes	Agreement	9
Shake no	Disagreement	15
Hand circle	Attention	16
Finger beckon	Invitation	21
Wave	Salutation	11
Fist pump	Affirmation	7
Hands to head	Frustration	7
Hand wave	Equivocation	23
Clap	Celebration	10
Open hands	Welcome	13

Table 3.1: A summary of the collected human gesture videos.

Figure 3.4: Screenshot of the identity experiment user interface.

3.4.4 Filters

We chose several different filters to assess with our experimental framework.

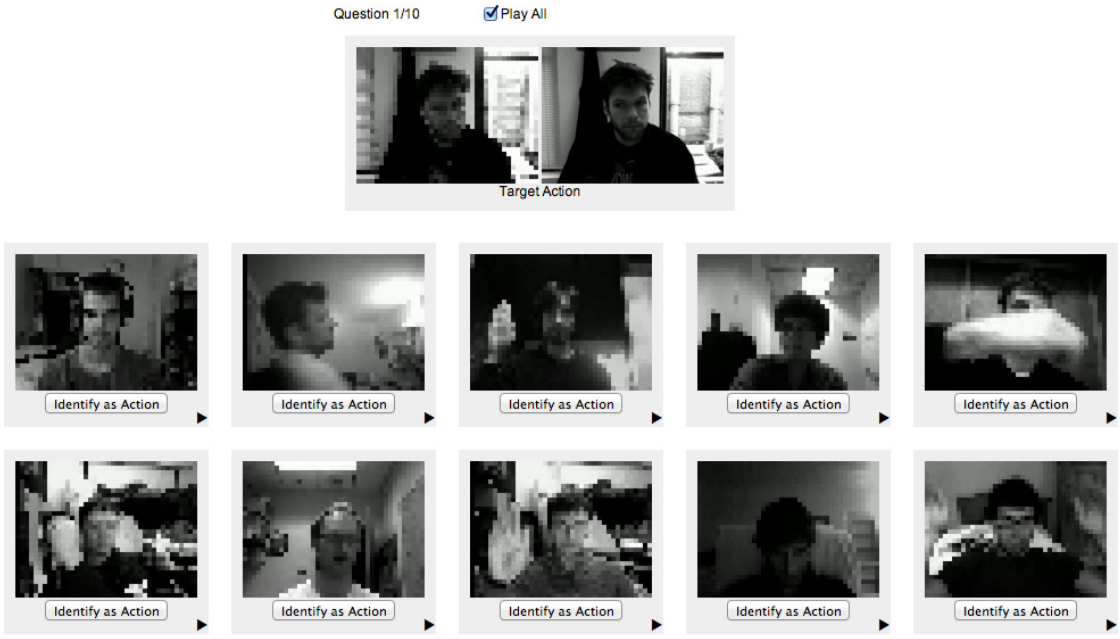

Figure 3.5: Screenshot of the gesture experiment user interface.

Pixelization

This filter is equivalent to reducing the resolution of an input video. The video is downsampled and scaled back up using nearest-neighbor sampling such that output is composed of larger pixel blocks. We used block sizes 4, 6, and 8.

Gaussian Blur

This filter applies a Gaussian blur to every frame of the input movie for a given kernel size. The kernel size determines the bluriness of the output movie. We used kernel widths 13, 25, and 37.

Canny Edge

The Canny edge detector [14] is useful for finding edges within an image and is thresholded such that output is binary. This means figures will tend to be outlined

but all subtleties of shading are lost.

Frame Differencing

Each pair of successive frames are subtracted. If there is no movement between two frames, the difference will be 0, a black image. Otherwise, the part of the image that changes will appear in shades of gray. If the overall lighting in a scene changes quickly, an inverted image of the scene can appear.

Greendot

This technique uses the Shi-Tomasi definition of strong image corners [72] to find good candidates for tracking. After these points are located, they are tracked in each subsequent frame using sparse optical flow [34]. Every 5 frames, additional Shi-Tomasi features are located and added into the pool for tracking. The result is a visualization of motion that is highly sparse. For example, flat (non-textured) regions will tend to be assigned very few features. This approach is inspired by work in pattern recognition where motion extraction is used as pre-process then fed into a classifier [99].

3.4.5 Experimental Setup

From our corpus of videos we generate random questions of either the gesture or identity type described above. For each question, we also choose a filter and apply that to the target and lineup videos. All filtering is done offline and in advance. We deployed the questions to a large pool of test subjects obtained via Amazon Mechanical Turk. The task requires no skills beyond non-impaired vision and no special software except the Flash plugin. Through our experience with Mechanical

HITs	Questions	Workers	Base Cost	Bonus Pay
400	4000	205	$100	$73

Table 3.2: A summary of our deployment.

Turk, we found it is better to batch many small tasks (HITs) into one larger task and pay an appropriately larger bounty. Even if the work pays the same effective hourly rate, higher paying tasks get more attention and are completed sooner. For the sake of this experiment, we group together 10 questions of a particular type and use a different filter for each question. For these 10 questions, we pay 25 cents. We want users to try hard to get the answer right so we offer and clearly state that a score of 100% will result in a bonus of 1 dollar, significantly higher than the base pay. We allow workers to complete each of the two experiment types twice, so a single user may contribute up to 40 assessments.

3.5 Results

We deployed 200 instances of each experiment type, as shown in 3.2. Figures 3.6 and 3.7 show the distribution of scores on each task where 1 represents getting 10 out of 10 questions correct. The shape of both graphs is bimodal. This suggests that there were two types of workers who completed our task: those who tried in earnest and those who clicked randomly despite our attempt to motivate them with bonus pay. The first mode in each graph occurs at 0.1 which corresponds to the expected value when someone clicks randomly. Looking into the actual clicks of the workers revealed that some always clicked the same button, confirming the suspicion. Given the design of our experiment, we should be able to remove certain data without biasing results since each set contains one question of every

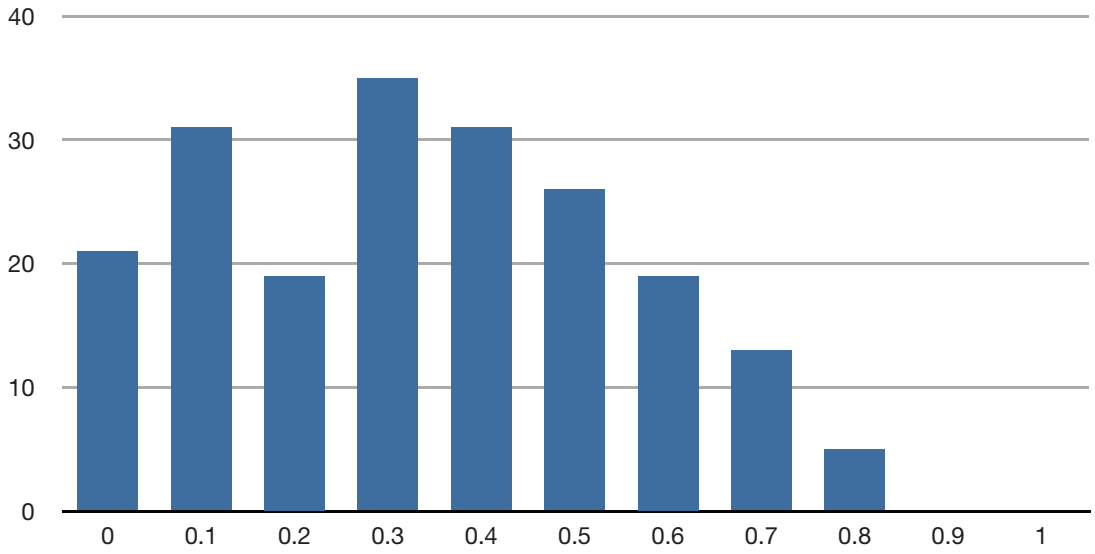

Figure 3.6: Distribution of scores on the identity experiment.

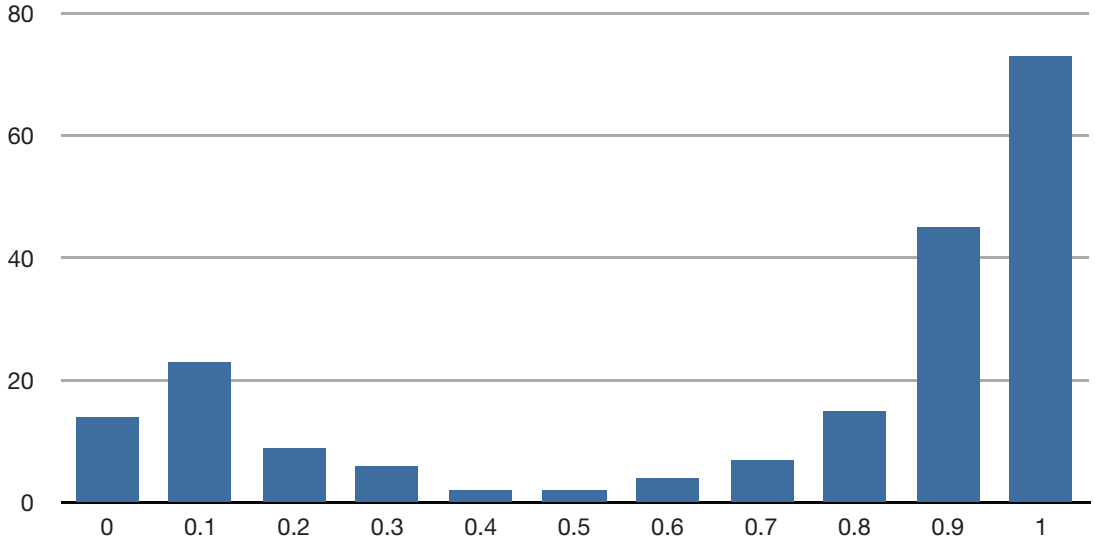

Figure 3.7: Distribution of scores on the gesture experiment.

filter type. In the case of the gesture experiment, excluding all work with a score below .4 will remove most of the noise due to random clicking. In the case of the

identity experiment, which is inherently harder, the two peaks overlap. We can exclude HITs with scores below .2 to reduce some of the noise. For our discussion we focus on the noise-reduced data but also include raw data for completeness.

3.5.1 Summary

Figure 3.8 show how test subjects did on the gesture experiments. We see that users did well on the gesture experiment across all filters. The accuracy ranges from 72% in the case of the heaviest pixelation up to 85% accuracy in the case of unfiltered video. We note that performance dropped as the strength of blur or pixelation increased, which confirms the quantitative findings in [8] and [54].

We see a much greater range in performance on the identity experiment across filters. Unfiltered video resulted in the best accuracy of 79%. Accuracy went down with the increased strength of the pixelation and blur filters as before. It is notable that the greendot filter had the worst identity performance at 15%, which is slightly better than chance.

3.6 Conclusion

The results of our experiments show the promise of the greendot filter in addressing our stated goal of reducing identifiability without removing body language. Furthermore, we show that video filters are not strictly zero-sum. Increasing the strength of the pixelation filter steadily reduces both the gesture and identity content of a video. But the greendot filter produces significantly lower identity recall with comparable gesture performance.

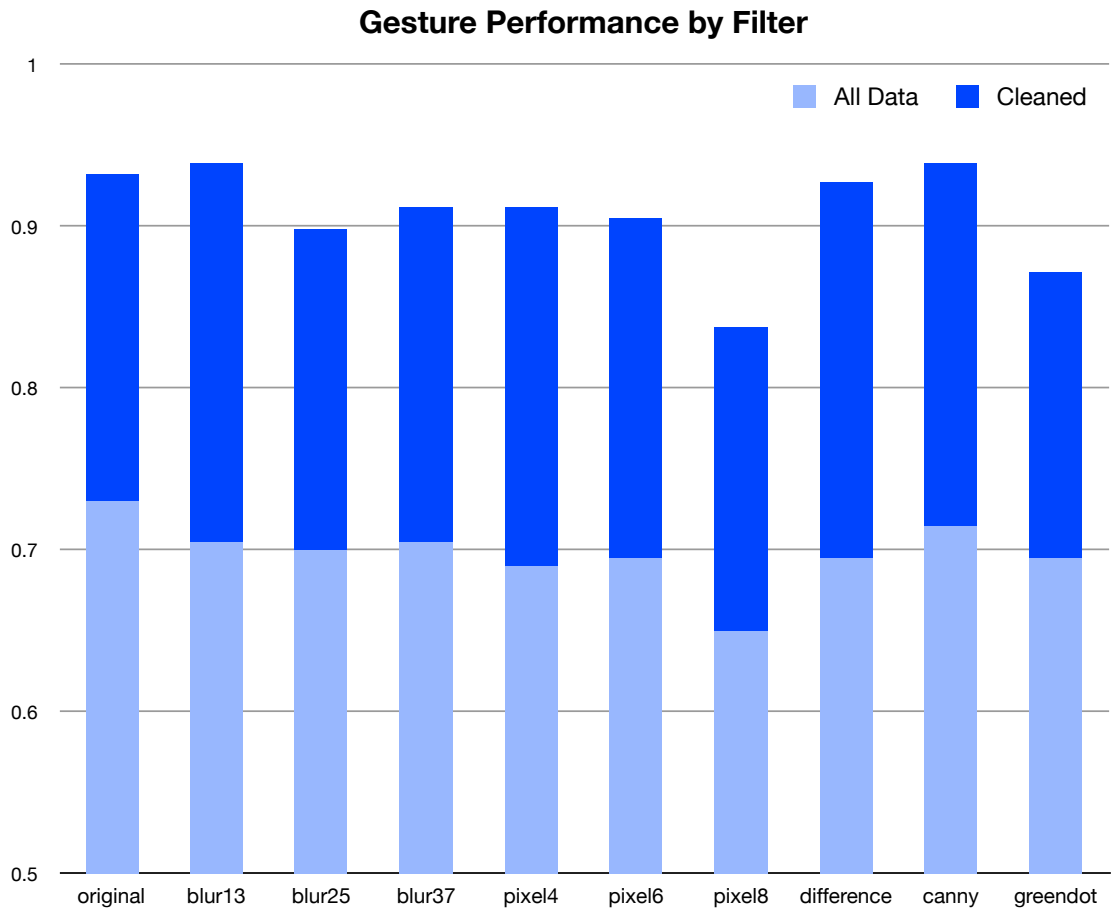

Figure 3.8: Performance of test subjects on the gesture experiment. The darker bars refer to data that has been cleaned to reduced random clicking.

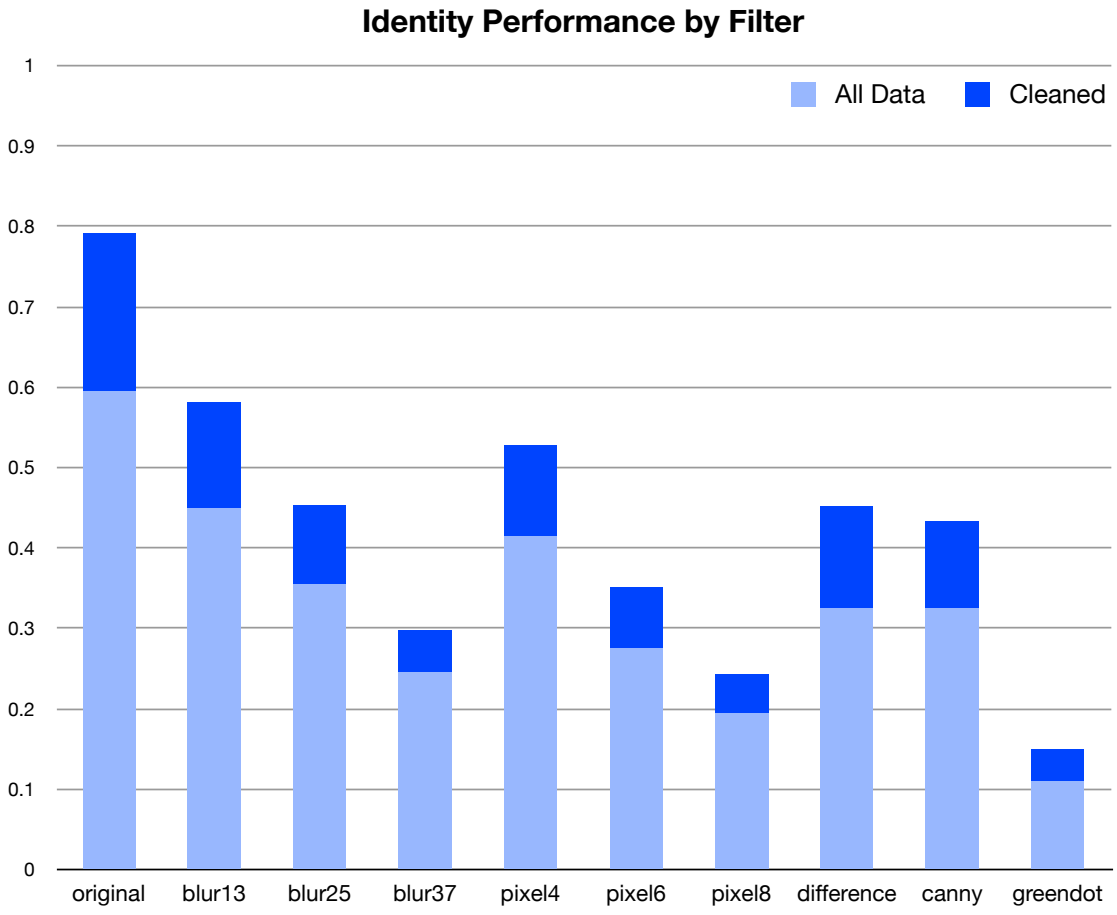

Figure 3.9: Performance of test subjects on the identity experiment.

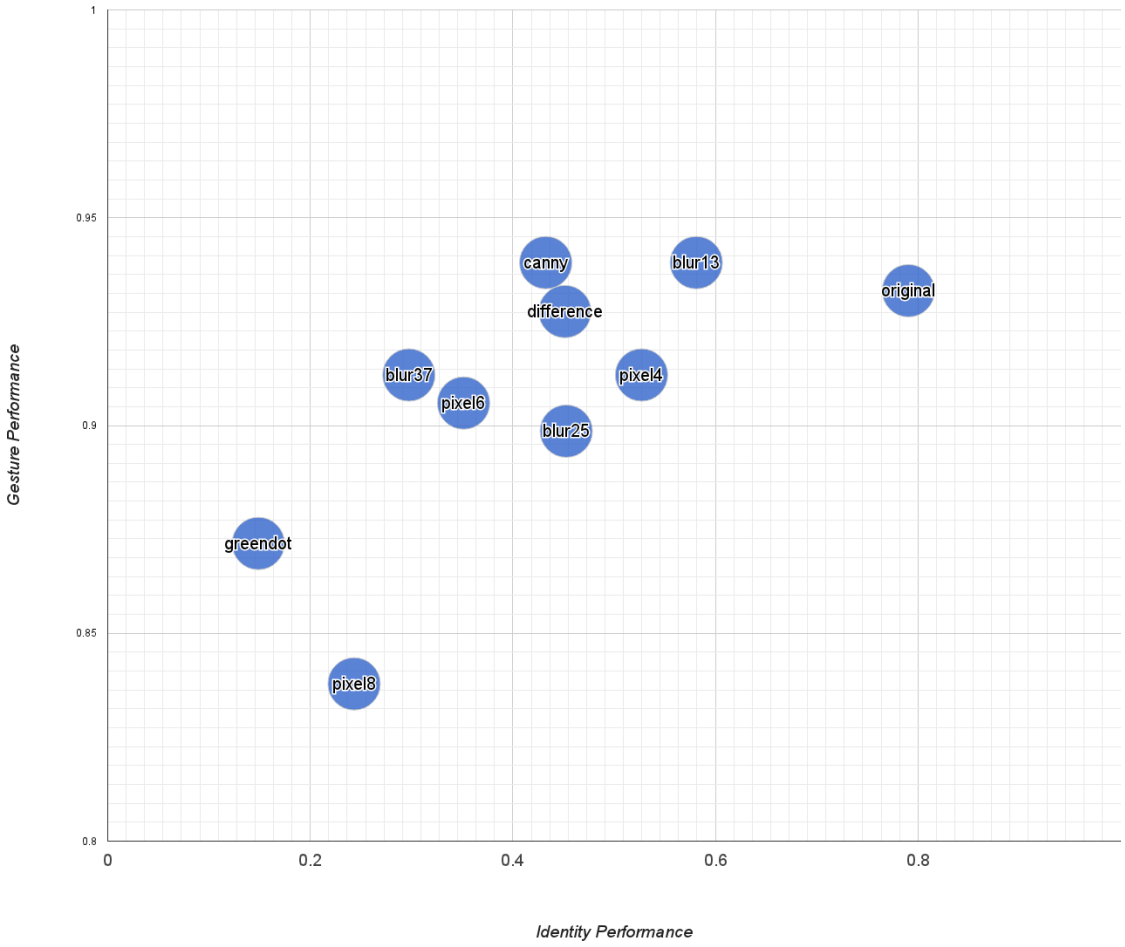

Figure 3.10: Identity vs. Gesture performance. (Cleaned data.)

Chapter 4

Motion Chain: A Webcam Game for Crowdsourcing Gesture Collection

4.1 Introduction

This chapter describes the development and preliminary design of a game with a purpose that attempts to build a corpus of useful and original videos of human motion. This content is intended for use in applications of machine learning and computer vision. The game, Motion Chain, encourages users to respond to text and video prompts by recording videos with a web camera. The game seeks to entertain not through an explicit achievement or point system but through the fun of performance and the discovery inherent in observing other players. This paper describes two specific forms of the game, Chains and Charades, and proposes future possibilities. The paper describes the phases of game design as well

as implementation details then discusses an approach for evaluating the game's effectiveness.

4.2 Motivation

The study of human motion is the underlying focus of the Movement Lab's work in computer vision. The traditional approach in this field involves building complex statistical models to automate the extraction of high level information from pixel data alone, or informally, to make computers that can see. Working in this area, we are constantly reminded of the incredible power and flexibility of the human perceptual system, as contrasted to the flaky performance of computer vision. Face detection is one example of a mature computer vision technology that works well; yet we can barely take the next step, to accurately identify and parse bodies. This means we are nowhere near what might be considered the holy grail of computer vision: to take as input any video clip and output a textual summary. We cannot reliably tag all objects and actors in a clip, let alone describe the relations between them and the relevant verbs in a video.

The vision community is working to develop ever more sophisticated algorithms to push the accuracy of automated visual tasks. One general class of approaches, called machine learning, takes examples and counterexamples which are fed into a model that "learns" such that it gets the right answer, described in [6]. At first, the model performs poorly, no better than chance; but with each additional example, accuracy may improve. The learning is successful when generalization is achieved, whereby never-before-seen input images can be correctly identified. A key ingredient for the vast majority of machine learning systems is the training

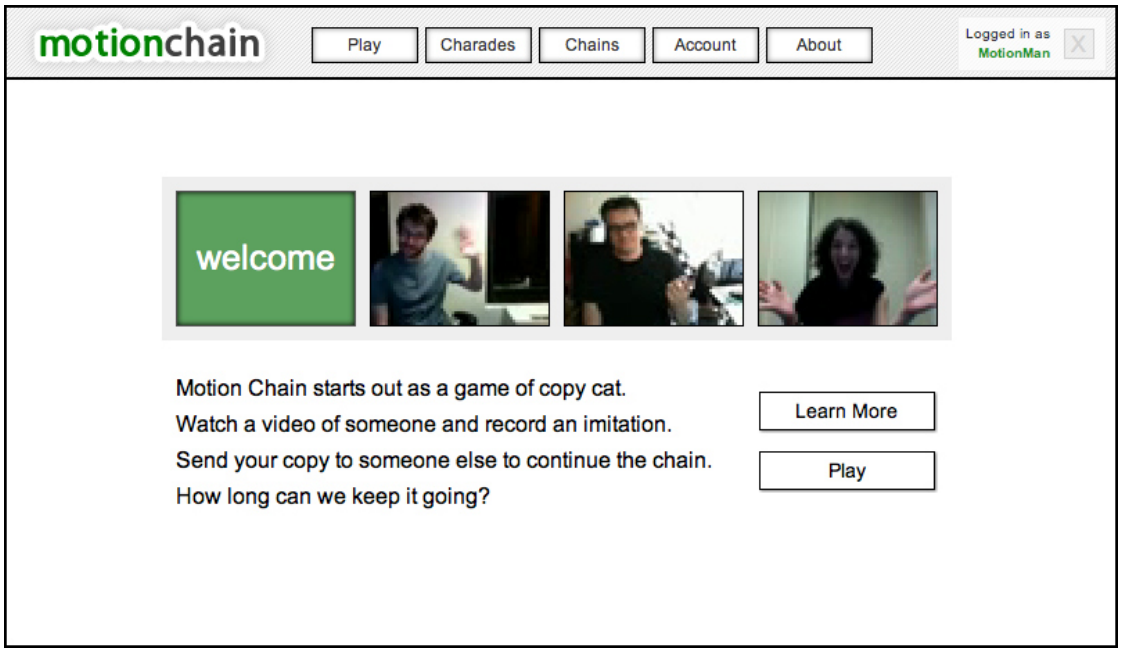

Figure 4.1: The Motion Chain website, located at http://www.motionchain.com.

data. And yet the research community tends to focus on the development of new models instead of data collection, which is treated as an abstraction. This project is an attempt to address the need for large datasets to drive machine learning solutions and to do so in the context of a game.

4.3 Related Work

One aspect of modern society, described by Clay Shirky in [73], is an overall increase in free time and more ways for people to spend it. Shirky explains that people are now motivated in their free time to do creative and useful things, whereas the previous era focused on the consumption of media. Examples include the public creation of Wikipedia, a collaborative effort to solve protein folding problems as in [42], and the detailed annotation of images and videos in [67].

These are examples of web-based crowdsourcing, wherein unpaid Internet users collaborate to achieve something good for society. More examples and a taxonomy of crowdsourcing can be found in [49].

The concept of Games with a Purpose, or GWAP, was pioneered by Luis von Ahn and described in [92]. The first GWAP, called The ESP Game, had users label images through a website. By adding competitive and collaborative elements to the game, people had fun in the course of performing a task that might otherwise be seen as work. The users were not paid for their efforts but elected to participate freely because the experience was fun and therefore intrinsically rewarding. In the case of The ESP Game, users correctly tagged millions of images which later helped power Google's image search.

Most crowdsourcing to date has involved the generation of labels or other text by users, though examples of more media-oriented systems exist. In [44], users were paid to draw pictures of sheep for an art experiment, while [51] had volunteers trace over frames of a recording to effectively rotoscope an entire music video. A video project found at [13] invited users to submit imitations of frames from a recorded video by using their web cameras, then combined these frames to create a unique visual experience. Our lab demonstrated in [85] that the resulting dataset could be used for novel machine learning applications.

4.4 Design Concept

The present work is an attempt to extend the scope of GWAPs to a richer class of user inputs. Typical GWAPs are used to obtain sparse information such as image labels. The input is rich, a collection of pixels roughly a megabyte in

Figure 4.2: A sample chain showing several players making a scratching gesture.

size, whereas the output is sparse, a short string smaller than a kilobyte. Motion Chain is a GWAP that motivates users to produce megabytes of rich output in the form of video.

The basic setup of Motion Chain is to let users interact with each other via short video clips and text. This is a subset of what can be done already on YouTube, but YouTube lacks the constraints or context to suggest using it for such a game. To develop my concept, we looked to parlor games of the Victorian era for inspiration. Such games rarely involve equipment or complex rules and gameplay is typically informal, with a focus on fun and creativity instead of points. Two such parlor games, Telephone and Charades, are directly adapted here.

4.4.1 Chains

In Telephone, one player whispers a phrase to someone else, who in turn, passes the message on. After several iterations, the phrase can morph into something completely different. Motion Chain takes this concept to the video domain. Players watch a short video clip of a person and attempt to copy it. The player can practice any number of times before recording and uploading a response. Next, the player is encouraged to send her clip to a friend, as specified by an email address. The friend receives an email encouraging him to play and he can, in turn, copy and send it on. Users are only allowed to see the full series of clips after submitting to that chain. The fun of the game lies in the discovery of how a gesture morphs over time. Another element of fun, a result of the web-based nature of the game, comes from discovering the other people who have participated in a chain, whether friends, acquaintances, or strangers.

4.4.2 Charades

This classic game asks players to convey words and phrases through mummery. Players take turns acting and guessing, which is made fun when hard prompts require lateral thinking and cleverness, both to act out and guess. A connection is formed between actor and guesser, as they have managed to communicate despite the no talking constraint. This web-based interpretation of Charades allows users to record clips and submit them to a public area of the site. Other players can see submitted charades and type in a guess, which also becomes public. So as not to spoil the answer, a player can't see the intended answer or other players' answers until she has submitted her own guess, which further motivates participation.

4.5 Design Process

The design process began with the development of a sandbox of code including form elements and Flash video units. We did not have a concrete idea of the gameplay but knew it would involve recording, playback, and text input. Once these components were complete, we could more rapidly prototype combinations in PHP and HTML and do informal user-testing on colleagues. We considered several prototypes at weekly lab meetings to see how others reacted and to prompt discussion. Once it was possible to record, we demonstrated it and had others try it. The most obvious thing to record was exactly what the first person had done, a clap. Watching several clap videos side by side was immediately compelling, and suggested the idea of a copying game as a starting point.

The second phase of development was to combine sandbox components and a database layer to create Chains. To launch the alpha test, we selected 40 friends

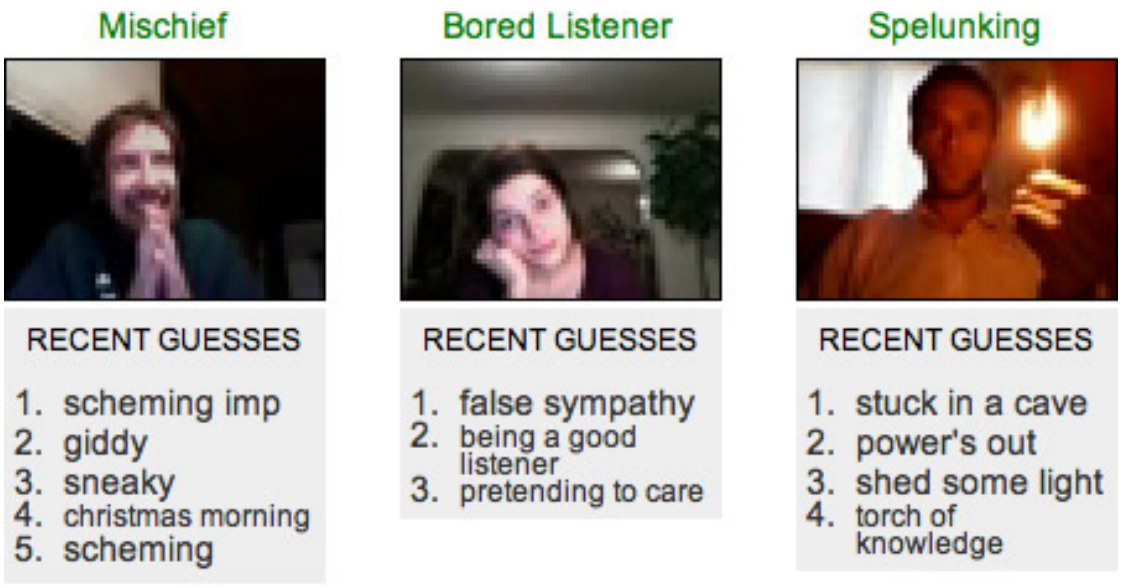

Figure 4.3: Some sample charades and recent guesses.

and colleagues and invited them to copy a variety of arbitrary gestures we happened to produce during the testing of the recording widget. Within days, the alpha testers collectively produced over 300 short video clips.

The third phase involved more serious engineering, such as the addition of proper user logins, web 2.0 features like voting, invites, reporting of inappropriate content, and the development of a points system. The intention of the points system is not to make the game more competitive but to help steer user behavior. If all users playing Chains want to create new chains but never want to copy other players, the game will not work. With the point infrastructure, we can reward players with a point for a copy, and charge some number of points to start a new chain. Likewise, we can offer points to encourage voting.

The nature of Motion Chain is collaborative and social so we recognize that the project will only succeed if it reaches some critical mass of users. Therefore current development is centered around social media integration and general promotion of

Figure 4.4: The recording interface. The previous recording is shown to the left of the live camera input.

the site. The aforementioned point system is potentially inhibitory to play, so until critical mass is reached, it is simply disabled. Another current focus of development is to collect a large set of starter content, such as example charades and chains. We can curate this content with the hope of creating a good experience for new users 'out of the box.'

4.6 Implementation

Motion Chain takes the form of a website, located at `http://www.motionchain.com`. The site was built on LAMP: Linux, Apache, MySQL, and PHP. These layers interact to route data between users and a database that tracks game state. Video files are stored remotely on the server, uploaded via a recording widget built in Flash. PHP acts as the glue to bind web server to database and implement most

of the control logic. PHP is both very flexible as a language and potentially hard to manage from the perspective of software engineering. To facilitate rapid prototyping with a strong development paradigm, we used a PHP framework called CodeIgniter. This simplified many coding tasks, such as the implementation of user accounts and password management. The web server for Motion Chain is located in the cloud, hosted on Amazon's EC2 service. The advantage of this service is the abstraction of hardware such that one can swap out one server for another with just a few clicks, simplifying upgrades if and when web traffic jumps.

The Flash recording widget required significant development. Flash remains the only way to reliably capture video from users on a variety of (non-mobile) platforms. The standard Flash solution for recording involves a remote media server and an adaptive approach to bandwidth allocation. For users with slow connections, the video quality goes down to prevent lag. For this project, upload time is less important than video quality so frames are captured to memory in the Flash plugin then uploaded to the server after recording is complete. Frames are uploaded individually as compressed JPEGs then stitched into a movie on the server. After making many videos with the widget, we realized the need for one special recording feature. When a user appears in a webcam preview on screen, it is natural for him to look at himself instead of the camera. Thus the user tends not to make eye contact with future viewers. The recording widget remedies this by moving the preview window up to the top-middle of the screen, closer to the most typical camera location, resulting in improved eye contact.

It's worth noting that the use of Flash, and thus personal computers (as opposed to mobile devices), results in a distinct and useful constraint on input. In this setup, the camera is nearly always fixed. This fixed camera helps many computer vision

tasks, since background subtraction can be employed and the video will tend to be less blurry than a similar video recorded on a mobile device.

4.7 Purpose

To use machine learning in service of motion-oriented computer vision tasks, we require example videos of people and corresponding text labels. If we had a thousand short videos of people performing a particular gesture, such as clapping, we would have a good chance of training a state-of-the-art clap detector. Previously, researchers might have paid hundreds of students to physically come to a lab to be recorded for such a purpose. Motion Chain has the potential to produce the same set of data, for free and in a short period of time. The games described above, Chains and Charades, are designed for the purpose of fun more than for science. We first have to obtain critical mass before we can get more specific with data collection; if we achieve a steady user base, we can introduce game prompts that are less immediately fun but more useful to our research agenda. The game could mix these prompts in with more fun ones, or try appealing to user altruism by explaining that certain prompts support scientific research.

4.8 Assessment

Our system needs to be nominally functional so as to facilitate various game scenarios, but fun itself is what users want, as opposed to a maximally efficient interface. With basic recording and playback in place, we can focus on developing the user experience and game scenarios. We recognize there could be many compelling gameplay modes beyond the two described above. The instruction for

Chains is "copy this motion" but that string can be changed to anything else. We hope new variants will emerge over time, as more users get deeper into the game. In the meantime, we can focus on producing and finding the most interesting videos for Chains and Charades. With the built-in logging tools, we can make estimates about the experience of users' playing habits, to study which game types and particular videos get the most views or responses and what patterns of sharing emerge. One priority is to study and increase the rate of obtaining new players. We can use word of mouth, social media, and online advertising to promote the site and recruit potential players but can't guarantee they'll stick around for more than a few seconds. Given the dynamic, data-driven nature of Motion Chain, it's possible to adapt the control logic to create A/B testing scenarios. When a user first comes to the site as a guest, she is shown sample chains and charades. We want to present the particular set of example videos that makes her most likely to take the next step and sign up for an account. Randomly selecting different sets of videos for different users can be correlated to the conversion rate for each group.

4.9 Discussion

In this chapter, we have described a new direction for Games with a Purpose that takes advantage of web-enabled social media, broadband connections, and the ubiquity of cameras. Using web cameras as an input for GWAP brings a new intimacy to games that could go deeper than interactions between icon-based avatars. Motion Chain is inspired by old parlor games, and yet has the potential to be something completely new. While the web creates an experience that lacks the immediacy of in-person gaming, it can produce a shareable trace of gameplay

and facilitate play between people separated in space and time. If the experience becomes so compelling that people play voluntarily and invite friends to join, Motion Chain will be a platform for creating great datasets to aid computer vision research.

Chapter 5

Cryptagram: Photo Privacy for Online Social Media

5.1 Overview

While Online Social Networks (OSNs) enable users to share photos easily, they also expose users to several privacy threats from both the OSNs and external entities. The current privacy controls on OSNs are far from adequate, resulting in inappropriate flows of information when users fail to understand their privacy settings or OSNs fail to implement policies correctly. OSNs may further complicate privacy expectations when they reserve the right to analyze uploaded photos using automated face identification techniques.

In this chapter, we propose the design, implementation and evaluation of Cryptagram, a system designed to enhance online photo privacy. Cryptagram enables users to convert photos into encrypted images, which the users upload to OSNs. Users directly manage access control to those photos via shared keys

that are independent of OSNs or other third parties. OSNs apply standard image transformations (JPEG compression) to all uploaded images so Cryptagram provides an image encoding and encryption mechanism that is tolerant to these transformations. Cryptagram guarantees that the recipient with the right credentials can completely retrieve the original image from the transformed version of the uploaded encrypted image while the OSN cannot infer the original image. Cryptagram's browser extension integrates seamlessly with preexisting OSNs, including Facebook and Google+, and at the time of publication, had approximately 400 users.

5.2 Introduction

Petabytes of imagery data have been posted by users to Online Social Networks (OSNs) with Facebook alone receiving over 250 million photo uploads per day [56], storing 10,000 times more photos than the Library of Congress [31]. Users feel the need internally and externally (peer pressure) to share photos on OSNs given the convenience of usage and their immense popularity [48, 7, 57, 59, 83]. Users share personal and potentially sensitive photos on OSNs, thereby exposing users to a wide range of privacy threats from external entities and the OSN itself [22, 81, 70]. We consider two basic factors that trigger privacy concerns for end-users in OSNs.

User/System Errors: A user who uploads an image to an OSN may wish to share it with only a select group of people, which OSNs partially satisfy with privacy settings. From the perspective of contextual integrity [55], the user is attempting to implement her own notion of appropriate information flows. But a recent study confirmed that Facebook users' impression of their sharing patterns

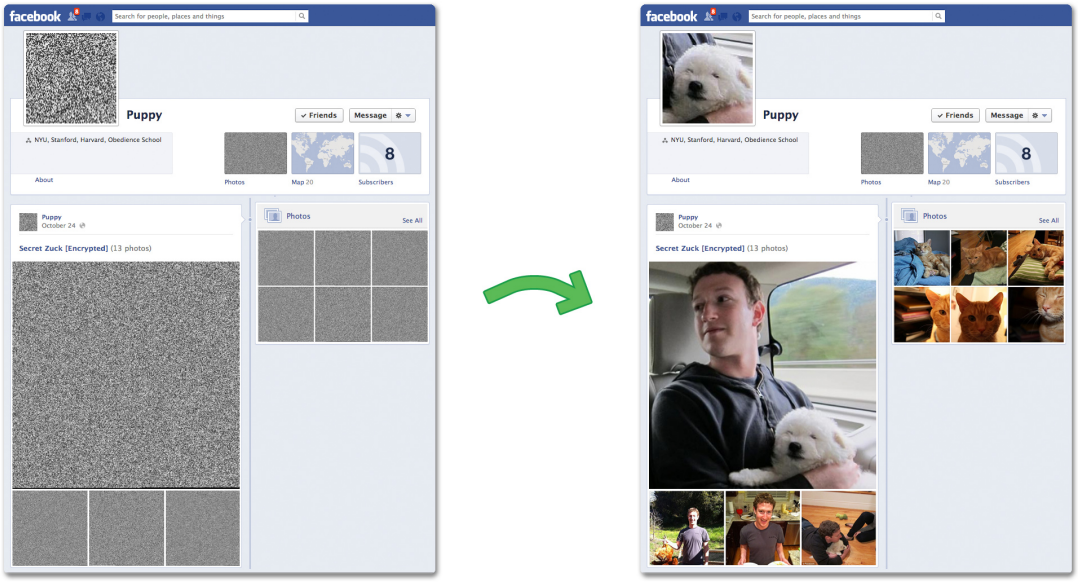

Figure 5.1: Example Cryptagram user experience. On the left, we show a social network with embedded Cryptagrams, uploaded by a user. A browser extension decrypts the images in place as shown on the right.

and their true privacy settings are often inconsistent [41]. Moreover, an OSN may fail to correctly enforce their privacy settings, such as the case when Facebook exposed its own CEO's private photos in a systemwide glitch [22].

Face Identification: A passive observer or a hosting OSN could extract large volumes of online photo uploads, indexing and discovering images within a corpus that belong to a specific user [81]. Mining of photo corpora can lead to the unexpected disclosure of individuals' locations or their participation in events. Facial data mining incidents have resulted in litigation against OSNs and further weakened the integrity of the relationship between the social network and the individual [70].

In this paper, we present the design, implementation and evaluation of Cryptagram, a system designed to address these photo privacy concerns in OSNs. A

basic design goal of Cryptagram is to build a usable solution that can offer strong privacy guarantees for end-users that remains backwards compatible with existing OSN user interface designs. To maintain the conventional feel of an OSN, Cryptagram uses the abstraction of an image interface (RGBA pixels) to manipulate the core image formats used by OSNs. Cryptagram leverages an end-to-end encryption system to transport images, which are uploaded to OSNs. Figure 5.1 illustrates a specific example of how Cryptagram represents normal images as encrypted images.[1]

A challenge in the design of such an end-to-end image encryption/decryption mechanism is to be resilient to image transformations by the OSN. For instance Facebook converts all uploaded photos, regardless of original format, to JPEG, choosing quality settings without user input. The recipient of a Cryptagram image must be able to retrieve the original image from the OSN-transformed version. In this paper, we describe the notion of q, p-Recoverability (Section 5.4.1) which formalizes the aforementioned property that enables the assessment of embedding protocol designs. We describe a class of JPEG embedding protocols that can achieve the q, p-Recoverability property for different JPEG quality transformation levels. The top-down techniques that we discuss for designing q, p-Recoverable protocols can also be applied to lossless image compression formats.

Cryptagram addresses a problem that is fundamentally different from conventional image steganography. While steganography aims to hide data in plainsight and *avoid detection* [28], Cryptagram makes obvious that it is hiding data with

[1]OSNs typically recompress and resize images within their backend infrastructure, presenting the most bandwidth-friendly version (thumbnails) as they deem appropriate. In order to render the decrypted Cryptagrams for thumbnails, Cryptagram infers from URL of the thumbnail how to fetch the full-size image, which Cryptagram fetches and decompresses when a user indicates our extension should do so.

the added aim of efficiently transporting bits in the image medium while being robust to image transformations. Despite the differences in problem definition, steganography does have the same mechanical use as Cryptagram for transporting bits in an image. When comparing the effective efficiency of our approach to steganography, Cryptagram packs many more bits per pixel (Section 5.7).

Cryptagram differs significantly from the recent work on photo privacy, P3 [61]. Unlike P3, Cryptagram operates completely in encrypted bit space and does not reveal sensitive cleartext data of photos to external entities (Section 5.9). Cryptagram also does not rely on third-party providers for providing photo privacy.

We present several key results in our evaluation. For JPEG Cryptagram images uploaded to Facebook, we find that JPEG compression quality for those high entropy images is in the range of 76 to 86 (for natural images, usually quality is 74). Given these recompression target ranges, we demonstrate JPEG embedding protocols that, in tandem with error-correcting codes, achieve an effective efficiency of 3.06 bits per pixel, which is 2.68× greater than the best related work. We also summarize a study of recoverability when recompressing already compressed images. We further illustrate a design point comparison of recoverability versus filesize expansion when comparing JPEG and webp lossy compression formats.

Our end-to-end Cryptagram system has been deployed to the web. Figure 5.2 summarizes a user's experience with the current decoder. Our decoder browser extensions integrate seamlessly with existing OSNs including Facebook and Google+ while being compatible with their complicated DOM structures. We have nearly 400 active users of our decoder extension and over 300 users have agreed to our IRB-approved study through which they submit high-level data about their ongoing image encrypting and decrypting habits with Cryptagram.

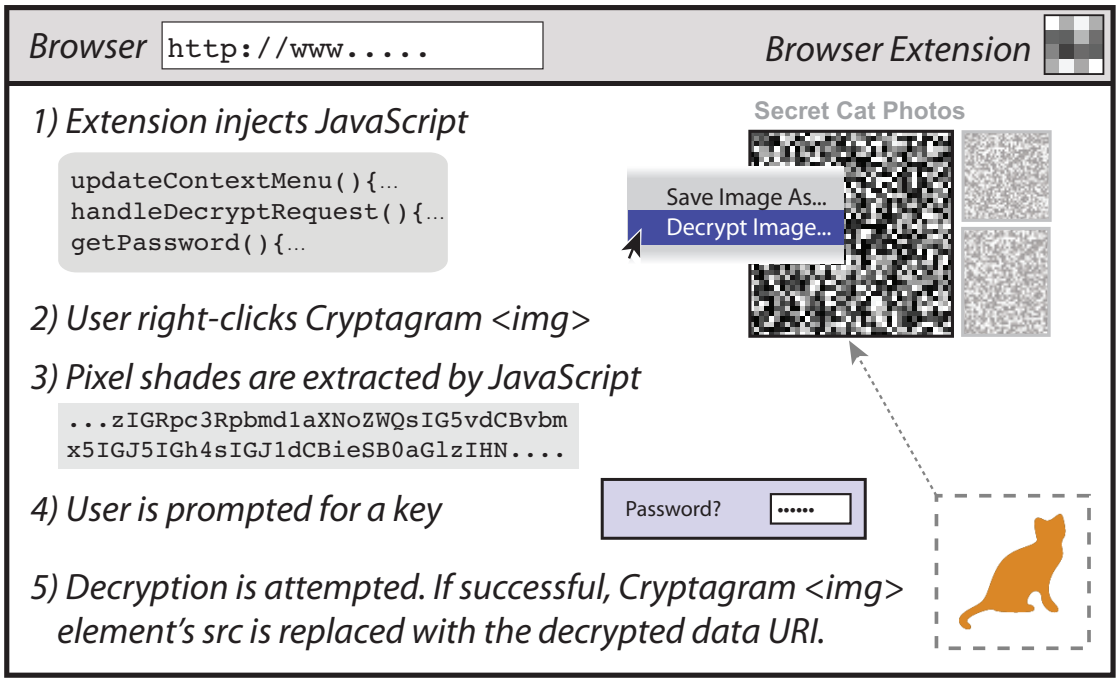

Figure 5.2: An overview of the Cryptagram user experience.

5.3 System Model

5.3.1 Problem Statement

The basic problem that Cryptagram aims to address can be stated as follows: Two users U and V are members in an OSN and have a shared key k (e.g., password), independent of the OSN. U wants to use the OSN to share an image I with V but does not intend to upload I to the OSN since the OSN or other unauthorized users may also be able to view I. Instead, U needs an encryption mechanism that can transform I into an encrypted image I', which V can retrieve, decrypt and obtain I (using the shared key k). The key challenge is that when U uploads an encrypted image I', the OSN can apply image transformations and the image V downloads may significantly differ from I'. Hence, the sharing mechanism needs

to be resilient to image transformations.

To better understand the transformation-resilient image encryption problem, we outline the basic image encoding and decoding steps used by Cryptagram and the property that Cryptagram aims to achieve:

• A user U encrypts a to-be-shared cleartext image, I, using a strong block cipher with a secret key, k, to produce a byte sequence, $E(I, k)$. This k may be human-readable (a password) or part of a hybrid cryptosystem in which the k is generated and shared using public key cryptography.

• An image encoding protocol, C, embeds $E(I, k)$ into the spatial domain of an image, $I_m = C(E(I, k))$ which the OSN transforms as $T(I_m)$. In this paper, we restrict T to be the identity transformation (lossless format) or a standard lossy image compression. We use JPEG for the lossy format in much of the evaluation since that is a commonly used standard across OSNs.

• An OSN user V who is an authorized recipient of the image needs to be aware of the block cipher secret key k. Using this key k and an image decoding algorithm, V should be able to successfully decode I from a transformed version $T(I_m)$. Here, we aim to achieve *recoverability* (intuitively, data integrity of the embedded message), which in the case of a lossless format is tautologically true sans other transformations. For JPEG, we aim for the q, p-Recoverability property: given a minimum quality level q of the transformed image $T(I_m)$, the decoding protocol should enable V to decode the original image I with high probability, p. We denote this recoverability probability using p, where the ideal case is when $p = 1$; however, achieving $p = 1$ may not always be feasible.

• Adversary, Eve, who passively observes $T(I_m)$ should not learn anything about I.

5.3.2 Design Goals

The problem statement highlights two key design goals of Cryptagram: ***data confidentiality*** and ***probabilistic data integrity for lossy images***. For data confidentiality, Cryptagram leverages trusted algorithms that users may use to ensure that data has been encoded with a strong proofs of security. For probabilistic data integrity, since Cryptagram aims to create images for use on OSNs, we can relax the constraint of traditional information security data integrity [24] for lossy image formats such as JPEG. This relaxation enables the Cryptagram design to incorporate features that demonstrate a spectrum of data integrity when a social network transforms uploaded images.

Usable: We aim to offer a system that affords users intuitive privacy on top of the images that they share on OSNs. While several offline approaches exist to preserve privacy (e.g., PGP [102]), Cryptagram makes it possible for users to create and share private photos without disrupting the OSN experience.

Widely Deployable and Applicable: To gain wide adoption, we have created a cross-browser, cross-platform, cross-image format system that enables Cryptagram to be used as both an encoder and decoder. The reduced friction to creating and accessing Cryptagram images removes the barrier to broader use of the technology.

Efficient: Compared to alternative methods, we present a system that offers significantly more data storage for a given file size or image dimensions.

5.3.3 Security Overview

Threat #1: Facial Data Mining

In this threat, the adversary is the OSN, whose aim is to execute facial data mining algorithms on uploaded images. In recent years, as social networks' corpi of images have grown dramatically, this is a serious concern for the privacy-conscious individual.

Approach. We have devised a scheme that reveals no information about the original image to the OSN. As we discuss in Section 5.5, the use of our embedding algorithm by nature thwarts facial data mining by embedding the cleartext (e.g., facial) data indirectly as encrypted bits in the spatial domain of the transport image.

Security Guarantees. With the use of a block cipher to transform the secret message, Cryptagram retains the strength of the underlying security properties of the chosen algorithm. With the use of public key cryptography users can retain cryptographic strength while leveraging a trusted, separate channel to bootstrap their sharing.

Threat #2: Misconfigured Privacy Controls

OSNs may fail to correctly deploy access control policies or users may accidentally misconfigure confusing access controls. The use of Cryptagram creates a separate channel of communication to ensure, with cryptographic strength, that only intended recipients see the cleartext photo. With the correct use of Cryptagram, an OSN could suffer a full system breach and encrypted images would remain private.

Limitations

Detecting and Blocking Cryptagram Images. Cryptagram does not address the problem of an OSN detecting and inhibiting the upload of all Cryptagram images. Steganography may be proposed in this scenario but problem redefinition and the tradeoff in efficiency make steganography an inappropriate application.

Unsupported Transformations. Though Cryptagram images are robust to varying degrees of JPEG compression, they does not support many other transformations. For example, cropping or rescaling a Cryptagram will generally break its encoding.

Brute-Force Cryptographic Attack. Cryptagram users who choose weak passwords in the symmetric key scheme can be attacked with dictionary or brute-force techniques. To address this limitation, we encourage users to abide by strong password creation techniques [50, 60] when using symmetric key credentials. When using public key cryptography, we encourage users to leverage the use of a public key infrastructure that is coupled with Cryptagram as we follow Key Continuity Management practices [30].

Copy and Paste. Users who gain access to cleartext images can copy and paste those images to whomever they choose. We believe this problem will persist despite any attempts, short of controlling all hardware at the disposal to humans accessing social networks.

5.4 Image Formats in OSNs

Several image formats are used across OSNs. While Facebook uses only the JPEG format to store images (and, moreover, strips uploaded images of EXIF

data), Google+ and other networks allow for a variety of lossless (e.g., PNG) and lossy (e.g., webp) formats. Our goal is to design a generic photo privacy solution that can work across different image formats. While lossless compression techniques are relatively easier to handle, determining an image encryption/decryption mechanism in the face of a lossy transformation is much more challenging. Given the popularity and broad use of JPEG, we use JPEG as our primary image format to describe the design of Cryptagram. We show how Cryptagram can be easily applied for other image formats including lossy image formats like webp.

Our design primarily focuses on embedding data in the spatial dimensions of an image. We define an embedding protocol to be an algorithm that describes the creation of a sequence of bits and how those bits are embedded into the spatial domain (pixels) of an image. We design embedding algorithms that work in a top-down fashion; that is, the data to be embedded is written into the spatial domain of an image on a pixel level rather than in any protocol-specific manner. We believe that a top-down approach allows us to meet the aim for wide deployablility and applicability in terms of implementation, testing and future image formats. The top-down API means that the design of codecs can apply or be tested across multiple formats with ease. When codec design depends on DCT coefficients, for instance, there are non-intuitive programming interfaces that would be required to make that facility addressable to the PNG format and not just JPEG, webp, and other DCT-coefficient based compression algorithms.

Assuming a passive adversary, this approach is a valid solution to the security threats that we outlined in the previous section. This is an especially prudent design choice considering that lossy image transformations will most intuitively aim to preserve higher order features of an image rather than its bit-wise representation.

The generic interface to the image is thus the four possible color channels red, green, blue, and alpha as well as their corresponding bit value. For JPEG, this means up to eight bits per the first three color channels. For PNG, we have up to 16 bits per channel for all four possible channels.

5.4.1 Defining q, p-Recoverability for JPEG

JPEG image transformations are inherently lossy in nature. With the aim of probabilistic data integrity, we make concrete the goal of relaxing the constraints of traditional notions of information security data integrity [24] for embedding data in JPEG.

We define the *q, p-JPEG Recoverability* (or, simply *q, p-Recoverability*) property of embedding protocols as follows: given a minimum quality level q that an OSN preserves in a transformation T of an uploaded image, an authorized recipient should be able to decode the original image with high probability p, where in the ideal case $p = 1$. The concept of q, p-Recoverability can also be applied to other lossy image transformations though the corresponding attainable values of q and p are dependent on transformation T.

In the context of JPEG images, we define a Cryptagram protocol as a message-JPEG encoder G and JPEG-message decoder G'. Given an input image[2] I, the first step in Cryptagram encoding is to convert the image into an encrypted sequence of bits $m = E(I, k)$, for clear-text image I and a block cipher key k. We refer to the input to the JPEG encoder as a sequence of bits m. Given m, the protocol encodes m in the spatial domain of a JPEG image, $I_m = G(m)$. JPEG (denoted

[2]We mean that I is a sequence of bits that represent an image format that a browser can render. Notably, Cryptagram's embedding can be a used with any arbitrary message I for delivering a message via the spatial domain of a transport image.

$$I_m = G(m)$$
$$G'(T'(T(I_m, q))) = m' =_p m \implies$$
$$G \text{ is } q, p\text{-Recoverable}$$

Figure 5.3: q, p-Recoverability in a nutshell.

by the function T, its inverse for decompression is T') compresses I_m at quality q to produce a sequence of bits, $T(I_m, q)$.

The recipient uses a two step decoding mechanism to retrieve an encrypted set of bits m': (a) the first step involves using the decompression step T' to produce $T'(T(I_m, q))$; (b) the second step involves using the JPEG-message decoder G' to retrieve an encrypted sequence of bits $m' = G'(T'(T(I_m, q)))$. Ideally, m' should match m; if they do, the recipient can use the secret key k to decrypt m' to retrieve the original input message. However given the lossy nature of the transformations, the message-JPEG encoding and JPEG-message decoding steps may not always succeed. Here, we use the term p to denote the probability that the algorithm successfully decodes the input bit sequence m. Mathematically, we denote this as: $m' =_p m$. If this constraint holds, then we define the protocol to be q, p-Recoverable. By considering a large sample set of input messages, we can statistically estimate the value of p for a given quality threshold q. The aim of Cryptagram is to identify q, p-Recoverable protocols that attain p close to one for low quality values and a high bits per pixel ratio. We summarize these ideas in Figure 5.3.

5.5 System Design

5.5.1 Lossy Images

To discuss how to embed data into a lossy image, we focus on the JPEG compression algorithm, though our design principles apply to other lossy formats.

How should one embed bits into the spatial domain of an image? To approach this challenge, we develop a mapping of bits to colors for specific pixels in an image. Intuitively, when choosing points (coordinates) in the color space to represent bits, we leverage the observation that the lossy codec may *shift* an initially embedded point (pixel's color) during encoding and decoding an image; however, the sphere in the color space within which that point may move does not overlap with other point-centered spheres. This is to say that when choosing what values to embed and how to coordinate pixel values, protocol designers must be sensitive to the assumption that the lossy codec will shift values within spheres in a color space. This intuition guides our JPEG design discussion below but, more importantly, is the generally applicable principle for Cryptagram protocol design.

The principal unit of embedding in Cryptagram is the Cryptagram pixel block (CPB). Multiple CPBs must fill or pack a 8×8 JPEG pixel block (JPB) for each channel of JPEG (luminance, chrominance red and chrominance blue), which is the "atomic unit" of pixels that undergoes JPEG compression [90]. We consider how to pack bits into the spatial domain of JPEG given two goals: *(1) efficient bit packing (increasing the number of bits per pixel)* and *(2)* q, p*-Recoverability.*

Embedding in the Spatial Domain

Cryptagram Pixel Blocks For protocol design we examine how to manipulate 64 pixels to embed bits efficiently. We embed symbols into the 64-pixel JPBs for each YC_bC_r channel with multiple Cryptagram pixel blocks (CPBs) per JPB. A CPB could be any shape that packs into a JPB. For our discussion, we consider 1×1 and 2×2 CPBs.

The composition of a CPB thus is a shape description, $w \times h$ (width, height) and a set of rules, R, for translating a symbol, x, or set of bits ($x = b_0, \ldots, b_{|x|}$) into red, green, and blue tuples (r, g, b) that we embed in each pixel of the CPB according to the appropriate $RGB \rightarrow YC_bC_r$ conversion. For simplicity, we represent the CPB embeddings for each channel as $L^{R_L}_{w \times h}$, where R_L are the rules that correspond to the channel, L, how to embed x to color values for L.

Because JPEG compression applies different downsampling and quantization matrices to luminance and chrominance channels (but applies the same compression to the two chrominance channels), we express the embedding protocol for a CPB as:

$$\left(Y^{R_Y}_{w_Y \times h_Y}, C^{R_C}_{w_C \times h_C}\right)$$

where Y corresponds to luminance and C to chrominance channels.

The rule set, R provides a large space for Cryptagram protocol designers. Intuitively, the composition of rules becomes a choice of three parameters: (1) how many bits to embed, (2) the number of discretizations to use (for which the number of bits to embed determines the lower-bound) in the color space, and (3) the choice of which discretization values from the color space to use. In short, we determine how many colors to use, what values they represent, and the resulting bitrate.

The JPEG compression algorithm compresses least the luminance channel of the three yielded by the $RGB \rightarrow Y'C_bC_r$ transformation. If we choose only to discretize values in luminance, we have an effective range of $[0, 255]$, which corresponds to grayscale. We denote this scenario as $\left(Y^{R_Y}_{w_Y \times h_Y}, C^0\right)$, using C^0 to denote that chrominance is not used.

On Chrominance Embedding. As described in the JPEG specification, the two chrominance channels are stored with significantly less fidelity than luminance. Both chrominance channels are down-sampled (by default in `libjpeg`, 2:1) and a more aggressive quantization table is used to further reduce the number of bits that need to be stored [90]. Intuitively, the chrominance channels are less efficient for embedding data in the spatial domain.

Encoding Algorithm We demonstrate the embedding algorithm in Figure 5.4. As we discuss the embedding algorithm at a high-level, we will refer to the concrete demonstration in that figure.

The first step of the encoding algorithm transforms the input clear-text image I into a sequence of encrypted bits m using a shared key k such that $m = E(I, k)$. Here, we use a standard block cipher algorithm AES in CCM mode (128 bit keys and 64 bit tag size). The encoding algorithm from this point chooses a small collection of bits at a time and converts these bits into Cryptagram pixel blocks. Figure 5.4 Step A shows how our example encrypted output message m is the sequence of characters, "Security." Using the base64 representation of the character, we know that the sequence of bits for each character is shown under Step B. We then show in Step C how the sequence of bits for the first character (S's representation as 010010) can be split into two three-bit sequences, the aforementioned

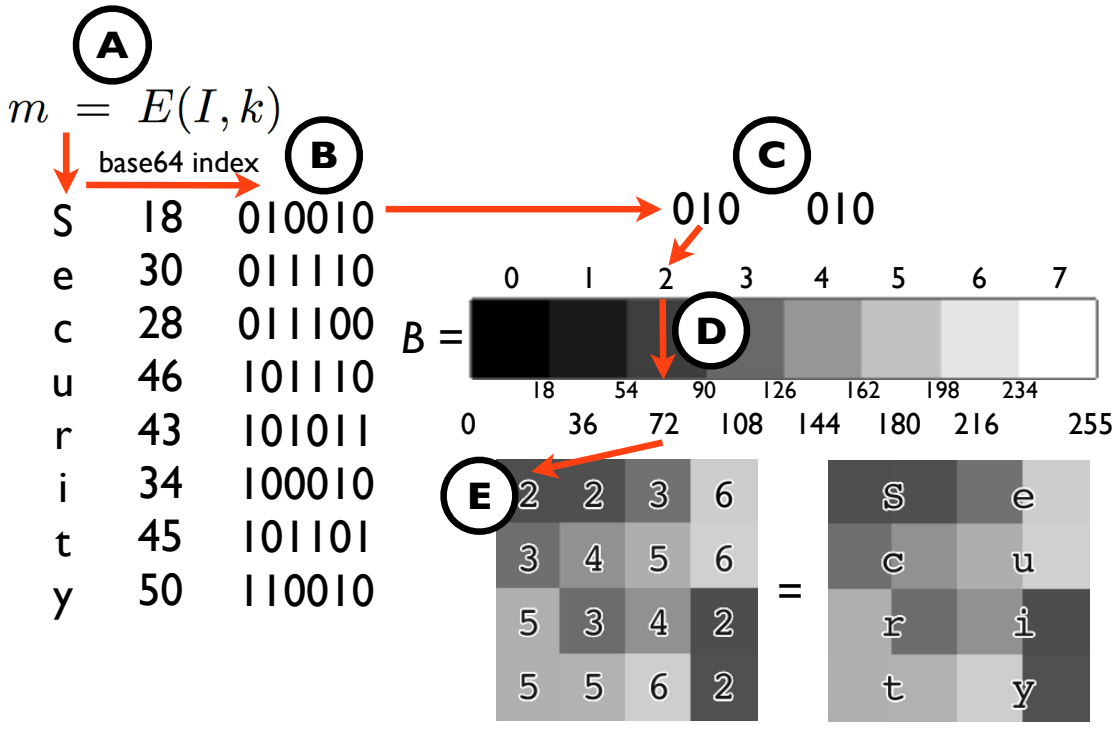

Figure 5.4: Encoding algorithm illustration. We demonstrate how Cryptagram maps an encrypted message's sequence of bits (Steps A-C) to color values (Step D) and how those correspond to embedded pixels (Step E).

"small collection of bits." Using a chunked grayscale color spectrum, we map the three-bits to an index in the array of values. The index's group representative (in this case at Step D, it's the grayscale value of 72) is what is embedded for the appropriate pixels, as shown in Step E. In this example, we continue to pluck off three bits at a time, for Steps B and C, then map those three bits values to grayscale values in Step D. Finally, we continue to embed the values left-to-right, top-to-bottom in this simple example for Step E. We have used a 2×2 CPB for this illustration, which packs perfectly into a standard 8×8 JPB. An alternative format could have used 1×1 CPBs, shading one pixel instead of four in Step E.

Figure 5.5 illustrates at a high-level where data is embedded in a typical Cryptagram image format.

We make the above example more concrete in the following formalism. An embedding algorithm, a, assumes that b bits will be embedded per CPB. Given b, a has a bijective mapping $B_L : \{0, 1, \ldots, 2^b\} \rightarrow L_a$ where $L_a \subseteq [0, 255]$. Given a bit sequence, s, a uses B_L to map b-length substrings of s to the corresponding L channel values that will be embedded at that particular pixel. Given the notation we have introduced, B is a more specific case of the notion of rule sets R_L we presented earlier. B_L mappings underpin the designs we present in this paper.

We can measure the efficiency of B_L based on b and the size of the channel's CPB to which the output of B_L mapped as $\frac{b}{|\text{CPB}|}$ bits per pixel, where $|\text{CPB}|$ is the number of pixels occupied by the CPB: $|\text{CPB}| = w \times h$.

From our discussion of the embedding protocol and q,p-Recoverability, we have laid the groundwork for how the designer's choice of protocol parameters ($d_{w,h}$, B, etc.) adjust the effective efficiency of the end-to-end protocol.

Example Encodings for Reasoning about q, p-Recoverability. To demon-

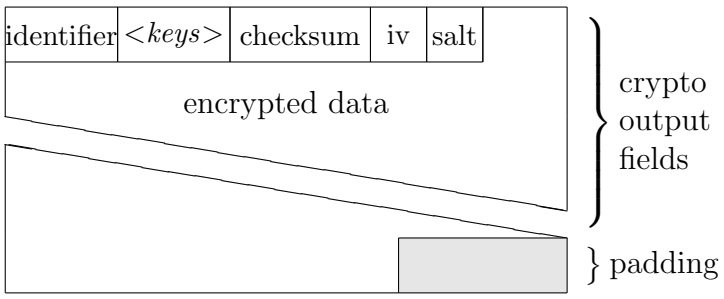

Figure 5.5: Layout of a Cryptagram. Each box is a sequence of shaded pixels representing the range of values for a particular protocol.

Name	**Notation**	**Luminance-only B Mapping**
Bin	$(Y^1_{1\times1}, C^0)$	$B : \{0, 1\} \rightarrow \{0, 255\}$
Quad	$(Y^2_{1\times1}, C^0)$	$B : \{0, 1, 2, 3\} \rightarrow \{0, 85, 170, 255\}$
Oct	$(Y^3_{1\times1}, C^0)$	$B : \{0, 1, ..., 7\} \rightarrow \{0, 36, 73, ..., 255\}$
Hex	$(Y^3_{1\times1}, C^0)$	$B : \{0, 1, ..., 15\} \rightarrow \{0, 17, 34, ..., 255\}$

Table 5.1: We present the B mappings for luminance-only embeddings in order to introduce the Y^n notation as well as illustrate the corresponding luminance values embedded in a Cryptagram using that mapping for the embedding protocol.

strate the tradeoff between efficiency (number of discretizations in B per CPB size) and q, p-Recoverability that we must consider in protocol design, we present two examples. The first example uses a $\left(Y^B_{1\times1}, C^0\right)$ CPB. As we translate (according to B) bits from m to successive CPBs color values, we fill the JPB from left-to-right, top-to-bottom, starting in the top-left of the 64 square pixel JPB, covering each channel independently. We can explore multiple color mappings B in order to see how q, p-Recoverability is affected by the $\left(Y^B_{1\times1}, C^0\right)$ CPB and B interaction.

We consider three mappings for B as shown in Table 5.1. The simplified representations for luminance will be used through this chapter. The superscript is the number of bits that can be embedded given the use of equally space values in $[0, 255]$, including extremal values.

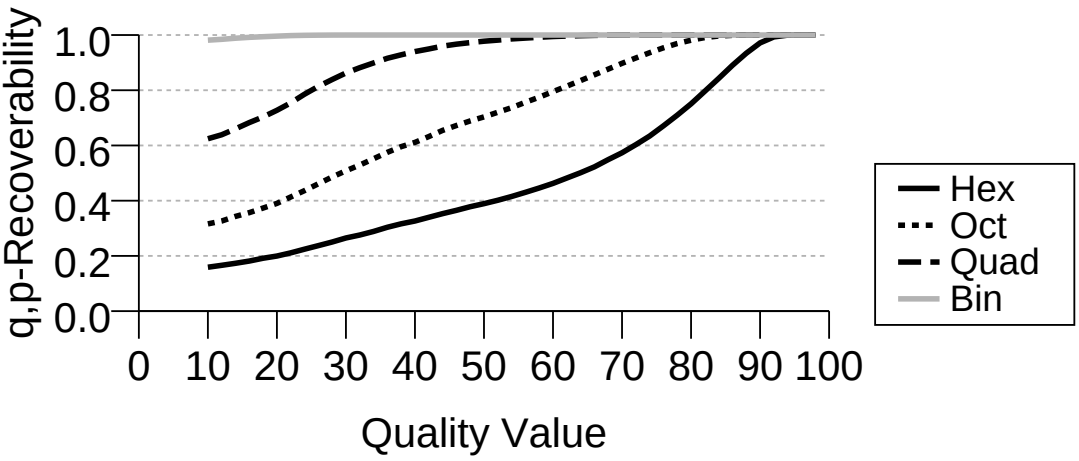

Figure 5.6: The relative performance of $\left(Y^B_{1\times1}, C^0\right)$ CPBs. We see that more discretizations results in weaker q, p-Recoverability as the quality to which we subject the JPEG to decreases. The tradeoff we must consider is what q, p-Recoverability we want to achieve (what minimum quality do we want a probabilistic guarantee) and how efficient we want for our embedding protocol.

Figure 5.6 illustrates the q, p-Recoverability of these choices. In comparing the best of binary, quadrature, octature, and hexature bits per pixel discretizations for the $\left(Y^B_{1\times1}, C^0\right)$ CPB, we have a sense of how the mapping choices perform relative to one another. Given a social network quality value (for JPEG recompression), we want to choose an embedding that allows for p very close to 1. If we choose the target quality to be 86%, then the values that are actually at $p = 1$ are the Quad and Bin mappings. These yield two and one bits per pixel, respectively. Because we are apt to choose a quality threshold conservatively, we opt to use the Bin approach: $\left(Y^1_{1\times1}, C^0\right)$.

Figure 5.7 shows the results of our exploration of the chrominance CPB size and the impact of embedding in luminance and chrominance concurrently. We must use 2×2 CPBs in chrominance channels to embed one bit per channel's block (or a cumulative 0.5 bits per pixel gain). We can thus embed in chrominance

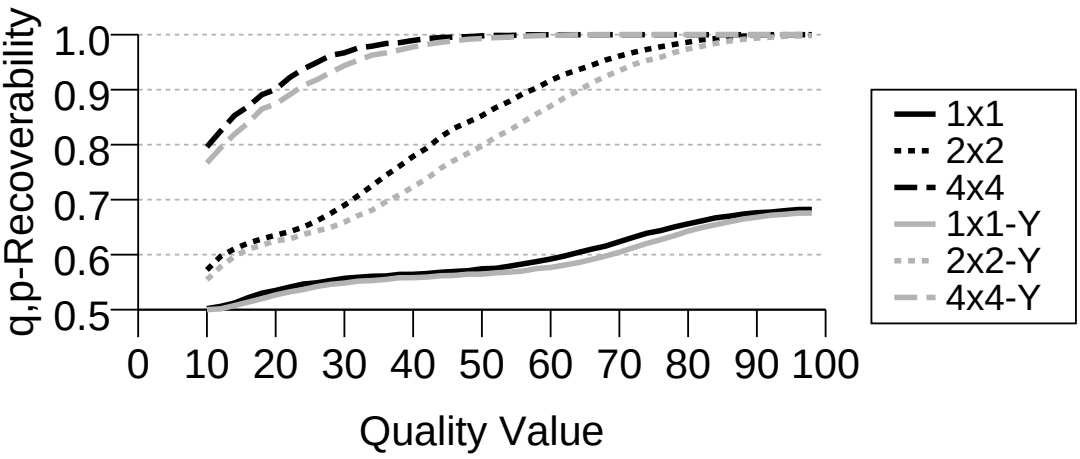

Figure 5.7: The feasibility of using chrominance to gain additional bits for embedding. All lines correspond to a chrominance B binary mapping. $n \times n$ corresponds to using only chrominance to embed bits using a binary mapping while keeping luminance set to 128. $n \times n$-Y embeds non-128 chrominance values along with chrominance.

as a function of the corresponding luminance values.[3] With this approach, we find that embedding more than two values per chrominance channel suffers low q, p-Recoverability. Thus while 4×4 appears to illustrate good q, p-Recoverability in the binary embedding case, the efficiency gain is so marginal as to be negligible. Thus we consider the of use $\left(Y^3_{1\times1}, C^1_{2\times2}\right)$ CPBs to gain an additional 0.5 bits per pixel. This gain with chrominance always requires error correction in order to attain q, p-Recoverability that is robust to OSN recompression.

On Decoding. To decode values from a Cryptagram JPEG, the decoder examines pixels in the same order as the encoder. Converting the RGB values to YC_bC_r, the decoding algorithm finds the nearest neighbor for the values in the co-domain of the B mapping for that protocol. The sequence of corresponding

[3]Notably, if the luminance values are at the extremes (0 or 255), then we do not embed a chrominance value in that pixel since no valid chrominance value exists.

domain values is the decoded sequence of bits.

Balancing q, p-Recoverability and Efficiency with Error Correction

Since the nature of protocols that we investigate are probabilistically recoverable (the p in q,p-Recoverability), we consider the use of Error Correcting Codes (ECC) in order to improve the q, p-Recoverability of our protocols while maintaining efficiency of the embedding algorithm. Reed-Solomon codes are of interest to us for their well-understood ECC properties and space efficiency. In our case and especially in Section 5.7, we use $RS(255, 223)$ protocol, in which we use the $x^8 + x^7 + x^2 + x + 1$, or "0x187", field generator polynomial and 32 bytes for parity in order to recover up to 16 byte errors for a 255 byte transmission. The input to $RS(255, 223)$ is the encrypted output of the block cipher algorithm. With the application of $RS(255, 223)$, the q, p-Recoverable protocol directly embeds the output bit stream from $RS(255, 223)$.

From Figure 5.7, we note how the use of chrominance would always require ECC in order to recover from OSN recompression-induced errors for the CPB case we have highlighted: $\left(Y^3_{1\times1}, C^1_{2\times2}\right)$.

5.5.2 Lossless Compatibility

Our effort focuses on the JPEG format given its online prevalence, but it's worth noting that our approach is seamlessly compatible with lossless formats such as PNG [21].

In this lossless scenario, we trivially achieve recoverability. The PNG format has a maximum per pixel efficiency of 64 bits per pixel. Each of the four channels, red, green, blue, and alpha, can store 16-bit values. We take 64 bits of sequential

data in m, split the 64 bits into four 16-bit segments, then write the respective 16-bit values into each of the four channels of a pixel.

5.5.3 Easy Key Management and Cryptography

Users have two options for managing access to their photos in Cryptagram: a symmetric key cryptosystem or a hybrid (symmetric key and public key) cryptosystem.

In the case of the symmetric key cryptosystem, Cryptagram makes sharing keys easy. A single key can be used for an image, set of images, or an album, and shared amongst a group of friends. This makes key sharing easy and manageable by design, and our Cryptagram browser extension facilitates the use of a password across multiple images or an album by allowing users to store the password. Enabling a strong password to be applied across an entire album of photos means that Cryptagram makes key dissemination easy.

Employing a hybrid cryptosystem by following the principles of *key continuity management* [30] means that the Cryptagram design focuses on guiding the user to use a hybrid cryptosystem correctly. In particular, by (1) limiting the interface for the use of public keys for encryption and private keys only for decryption and (2) using strong defaults for the block cipher and public key cryptography algorithms, Cryptagram reduces the friction to secure and correct use of a hybrid cryptosystem.

For both schemes, users do not share the sensitive information through the social network. We advise users to use a separate channel (e.g., text messaging) to share sensitive credentials (e.g., an album password) so as to conform to the threat model in which Cryptagram is designed to protect users from a hosting OSN.

5.5.4 Usable Image Identifiers

By default, all Cryptagram images appear as indistinguishable noisy images. We describe how we enable users to create images that are easier to identify for fellow human users.

Text Watermark: One challenge with the current format is that all output images look virtually identical. This is a problem when, for example, a user asks a friend for a Cryptagram password. Without a file name or album name, there is no codified way to refer to images. Using a simple extension to the encoding tool, we can enable the user to specify a text or image-based watermark to render underneath the Cryptagram image. A text watermark could specify useful identifiers, such as a URL or an email address for submitting password requests.

Chrominance Watermark: In cases that we do not use the chrominance channels for data transmission, we can use these channels for identification purposes. We modify the C_b and C_r channels to add chrominance to output Cryptagrams and do so without corrupting the luminance payload.

We embed images in these chrominance channels so long as luminance remains unaffected. This watermark is not suitable for embedding most natural images since, perceptually, we rely heavily on luminance, but the technique works well with high-saturation icons or logos.

5.5.5 Surviving Partial Failures

The current protocol has error correction and can withstand some degree of noise from JPEG but will fail with a cropping transformation. We can extend the basic design to provide cropping robustness by dividing a Cryptagram's payload

(a) Quality 78 (b) Quality 74 (c) Quality 70

Figure 5.8: A block-oriented failure resiliant protocol.

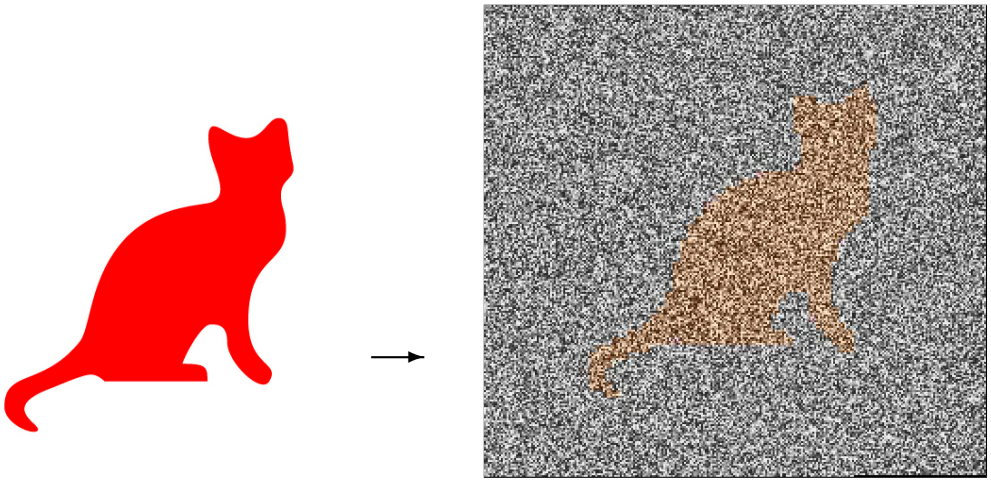

Figure 5.9: Example of a chrominance watermark. The purpose is to help users identify encrypted images, which would otherwise all look the same. (Note that black and white printing renders the sample watermark unnoticeable since the embedding essentially ignores the luminance values of the watermark image.)

into smaller units. We encrypt each block with the same password and decryption will involve individually decrypting and concatenating all blocks. If one block fails to decode correctly due to cropping, the integrity of other blocks and their sequence within the original JPEG remain unharmed. Such an approach, however, does not apply to storing arbitrary bit streams, but for images one can replace unrecoverable blocks with zeroes in order to display as much of the original image as possible as shown in Figure 5.8.

(a) (b)

Figure 5.10: Future Cryptagram features. Figure 5.10a demonstrates the use of the textual watermarking to enable users to embed textual messages in their images. Figure 5.10b shows the effects of using a partial image encryption scheme, which will require Cryptagram image detection.

5.6 Implementation and Deployment

In this section we describe the current state of the applications deployed under the Cryptagram name, including several components and continuously evolving inner protocols. The code is open source and online:

`http://github.com/prglab/cryptagram`.

5.6.1 Microbenchmarks

As of the publication of this thesis, the Cryptagram Chrome extension had approximately 400 users, according to the Chrome Store. We requested user-consent for an IRB-approved study to gather non-identifying log reports and consent has been granted from 373 unique browser installations.

We built the Chrome Extension with the Closure framework [32], requiring approximately 4000 Source Lines of Code (SLOC) [19, 96], porting the core components to a Firefox add-on with some additional code.

Our benchmarking framework consists of an ECC implementation and benchmarking code, and relies on the Reed-Solomon kernel module code ported to userspace, libjpeg-turbo codec, and a corresponding image interface ported from the Google Chromium browser [87]) (3000 SLOC).

The iOS App uses WebKit's JavaScriptCore to leverage the same cryptographic library as our JavaScript extension whereas the Android App achieves JavaScript integration through a WebKitView (2300 SLOC). These applications enable local Cryptagram encoding and decoding, without OSN integration. With respect to engineering we have found that configuring native cryptographic libraries across languages and platforms can be difficult. Wrapping JavaScript libraries results in a performance penalty but simplifies the assurance of algorthmic parity across platforms.

5.6.2 Browser Extension Decoder

We implemented the first version of the software as a browser extension, a framework supported by Chrome and Firefox browsers, which allows us to augment the user experience on any website by JavaScript injection.

For our first deployment of Cryptagram we adopted an embedding protocol with a $\left(Y^3_{2\times2}, C^0\right)$ CPB. This protocol also embeds a checksum for verifying the integrity of the decoded encrypted bits. The checksum is not of the cleartext data; it is a checksum of the encrypted data and embedded adjacent to the encrypted data for data integrity purposes.

Decoding in place. Extensions can access pixel data of images on a website. The extensions perform image processing to produce new images to insert into the original image's container, as shown in figure 5.2. We add a contextual menu item

so a user can right-click any image and attempt to decrypt it as a Cryptagram. With the correct credentials, the extension decrypts the original image, which pops into place.

5.6.3 Web-based Encoder

We wrote the web-based encoder in JavaScript with the Closure Framework, sharing much of the codebase with the decoder extensions. The encoder allows users to drag-and-drop cleartext images onto the encoder page. The drag-and-drop triggers an event to prompt users for a strong password (in the symmetric key case) as well as desired settings (e.g., the preferred tradeoff of a high-resolution, low-quality image or low-resolution, high-quality image). The encryption, encoding, and image zipping requires no server interaction and thus allows for complete offline operation by end-users.

5.7 Evaluation

We now explore the evaluation of the Cryptagram system. We begin with microbenchmarks as well as observations that serve as background for the subsequent evaluations. In particular, we will present the efficiency performance of protocols that we find to be the most useful for end-users and reason about the utility of the current deployment.

5.7.1 Efficiency Microbenchmarks

With microbenchmarks, we aim to establish a sense of the tangible weight that Cryptagram adds to the user experience of sharing photos as well as the system

overhead.

On Browser Performance. We found that the input file size to the encoder and decoder correlated linearly with time to execute. The approximate ratio of time to complete the operation (milliseconds) to input filesize (KB) was 2.684 for the encoder and 1.989 for the decoder on an iMac with 2 x 2.26 Quad Core processors in the Chrome browser (one core for the browser process). While the noticeable human visual reaction time is in the range of 190 to 330 milliseconds [45], the results demonstrate that the overhead of using Cryptagram for viewing OSN photos is marginal.

On File Size. Since the high entropy of Cryptagram counteracts the compressive power of JPEG, the output file size depends entirely on the chosen embedding protocol and constraints imposed by the OSN. For Google+ and Facebook, uploading images have a cap based solely on image dimensions. The authors have found that the maximum upload dimensions in these OSNs is 2048×2048. This means that for a scheme that attains an efficiency of three bits per pixel, we can store at most 1.5 MB in the spatial domain of an uploaded JPEG image.

How does the size of the input data relate to the output Cryptagram image size? The nature of JPEG compression complicates this question. The output Cryptagram image may be saved at 100% quality, creating a large filesize footprint. While this may seem necessary given that we examine q, p-Recoverability with respect to the compression applied by an OSN, the composition of JPEG compression is neither idempotent nor cleanly-defined recursively. Instead, as we explore later in this section, we consider the observed error rates of compressing already-compressed Cryptagram images (simulating what an OSN would do).

Table 5.2 shows the expansion ratio from a given input size. For the case of

	$(Y^3_{1\times1}, C^0)$			$(Y^1_{1\times1}, C^0)$		
Quality:	90	80	70	90	70	50
Expansion:	2.25	1.75	1.40	7.82	5.29	4.39

Table 5.2: We present the tabular data that illustrates the file size expansion when using various protocol choices in the Cryptagram framework.

a $(Y^3_{1\times1}, C^0)$ CPB with an output Cryptagram image with JPEG compression 80, the filesize on disk inflation is $1.75\times$.

For the sake of minimizing upload bandwidth, users may opt to export Cryptagram images with less than 100% quality and Cryptagram will still guarantee q, p-Recoverability within a certain range.

5.7.2 Compressing the Compressed

Apropos to the question of file size expansion, we examine the implications of a recompressed Cryptagram JPEG on q, p-Recoverability. Figure 5.11 shows the effects of exporting a Cryptagram to a JPEG Quality 1 and then (as an OSN would do) recompressing the image at JPEG Quality 2. The error rate indicates the fraction of CPBs that were broken through successive recompression. This data indicates that we can export Cryptagram JPEGs to 82% quality and OSNs' recompression still permits recoverability, assuming that we leverage $RS(255, 223)$ ECC.

5.7.3 OSN Photo Quality

As much of our evaluation relates error rates to JPEG quality level, we want to know the JPEG settings employed by popular OSNs. To estimate these quality levels, we exported a variety of images as quality 95 JPEGs, uploaded those images

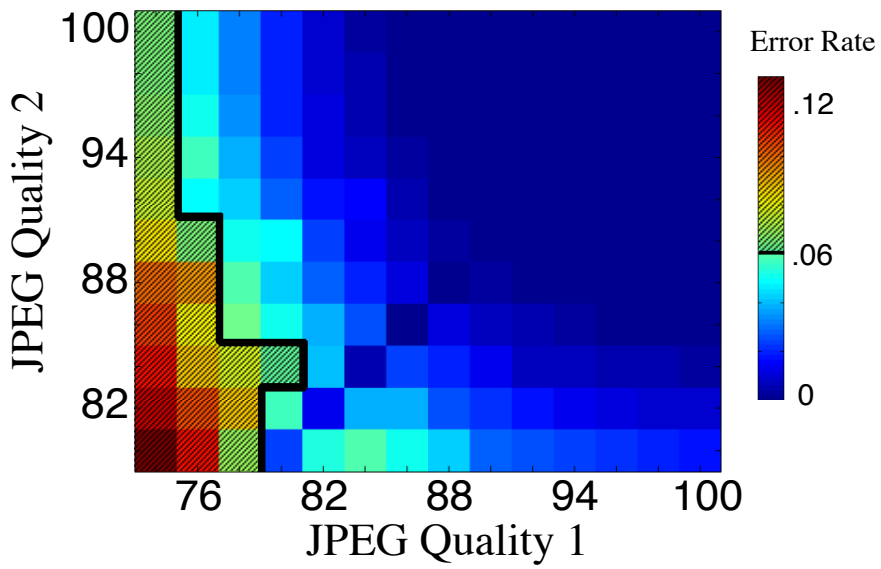

Figure 5.11: The effects of recompressing a compressed JPEG. The x-axis shows the quality of the original Cryptagram JPEG. The y-axis shows the recompressed quality of the JPEG. The line separating striped versus unstriped values is the q, p-Recoverability threshold we encounter with $RS(255, 223)$. Any values to the right of the 0.06 line show the successive recompressions that Cryptagram can tolerate for $\left(Y^3_{1\times1}, C^0\right)$. Error rates were determined by testing approximately 2,400 pseudorandom 8×8 images at each combination of quality levels.

to both Facebook and Google+, then re-downloaded the images for analysis.

On Google+, 30 such test images came back bitwise identical, meaning images were not recompressed.[4]

Facebook, on the other hand, applies JPEG compression to save disk space. After downloading images from Facebook, we looked for evidence of quality in the JPEG headers. Out of 30 natural images, 25 came back with a JPEG quantization matrix exactly equivalent to that of a quality 74 JPEG, the other five having matrices equivalent to JPEG qualities in the range of 76 to 86.

Fortunately for Cryptagram, high entropy images all appear similarly to the

[4]Google+ does recompress images for quicker display during album browsing but it is trivial to convert any such hyperlinks to their full-resolution equivalents.

JPEG algorithm and are treated predictably when uploaded to Facebook. All test Cryptagrams uploaded then downloaded came back with the quantization matrix from a quality 85 or 87 JPEG, which we measured by explicitly examining the quantization tables of the downloaded JPEG file. This quality level puts us safely above the necessary threshold of our deployed embedding protocol.

5.7.4 Embeddings with ECC

In this section, we examine the benefit of using ECC to reconcile the trade-offs we must consider between efficiency and q-p-Recoverability. We presented in Section 5.5 the performance of the $\left(Y^B_{1\times1}, C^0\right)$ CPB for various B mappings. From that experience, we conclude that a protocol without error correction is limited to using Quad or Bin mapping strategies.

We examine the utility of applying our ECC algorithm of choice for embedding data to measure q, p-Recoverability in lower quality regimes of JPEG compression. With the use of $RS(255, 223)$ for ECC, we note that we embed 14% extra data for the recovery so our subsequent evaluation considers the effective efficiency of a system that adds this data overhead.

Figure 5.12 allows us to explore the design space of applying ECC to evaluate the q, p-Recoverability for given $\left(Y^B_{1\times1}, C^0\right)$ luminance-only embedding schemes. We see that the Bin, Quad and Oct embedding schemes perform above p=94 in the regime around 85%, thus enabling us to achieve q, p-Recoverability on Facebook.

Figure 5.13 illustrates the benefit of using luminance and chrominance embeddings to achieve 3.5 bits per pixel embedding efficiency for q-p-Recoverability that satisfies OSN recompression and ECC. In the interest of saving space, we do not show the q-p-Recoverability curves, but instead summarize the details relevant to

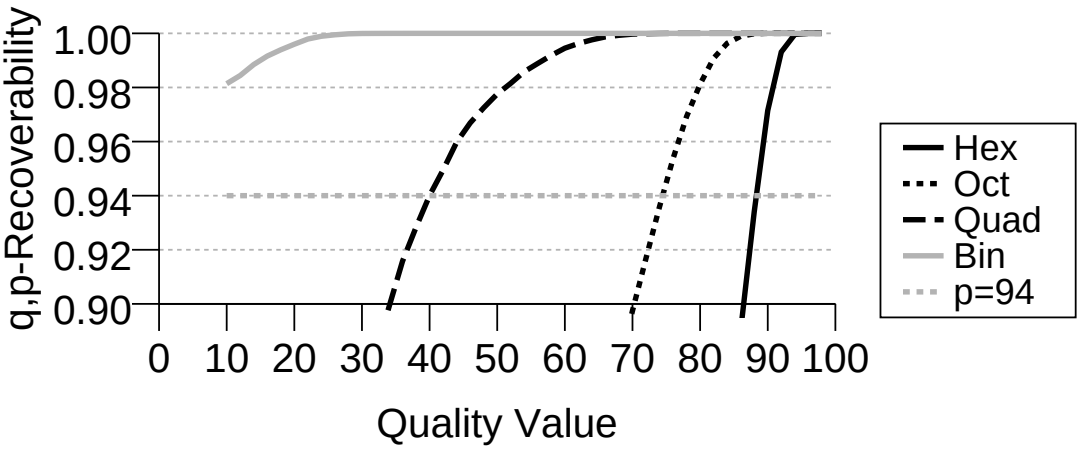

Figure 5.12: This indicates to us the feasibility of leveraging $RS(255, 223)$ to improve the q, p-Recoverability with various $\left(Y^B_{1\times1}, C^0\right)$ embedding protocols.

	Without ECC		With ECC ($RS(255, 223)$)				
$\lvert B\rvert$	$q, 100$-Rec (quality)	Efficiency bits/pix	$q, 94$-Rec (quality)	Effective bits/pix	Lum+Chrom $q, 94$-Rec	Efficiency bits/pix	Effective bits/pix
Hex	-	-	90	3.5	90	4.5	3.94
Oct	90	3	76	2.62	77	3.5	3.06
Quad	80	2	44	1.75	66	2.5	2.19
Bin	< 20	1	< 10	0.87	38	1.5	1.31

Table 5.3: Summary of the results that inform how to proceed with applying $RS(255, 223)$ FEC for embedding values in JPEGs that are recompressible.

the ECC discussion in Table 5.3. Given that ECC with $RS(255, 223)$ recovers up to 16 bytes ($\approx 6.27\%$) of damaged bytes for every 255 bytes of data, we can establish our target recoverability probability at $\approx 94\%$; in other words, if less than 6% of bytes break then applying ECC enables us to use that particular encoding scheme. We highlight in Table 5.3 the q, p-Recoverable protocol that we choose for Cryptagram.

This efficiency is superior to X-pire! [4], which had a capacity of two bits per pixel with ECC. We have $1.75\times$ this capacity, significant considering the size and quality of images this enables users to upload to OSNs.

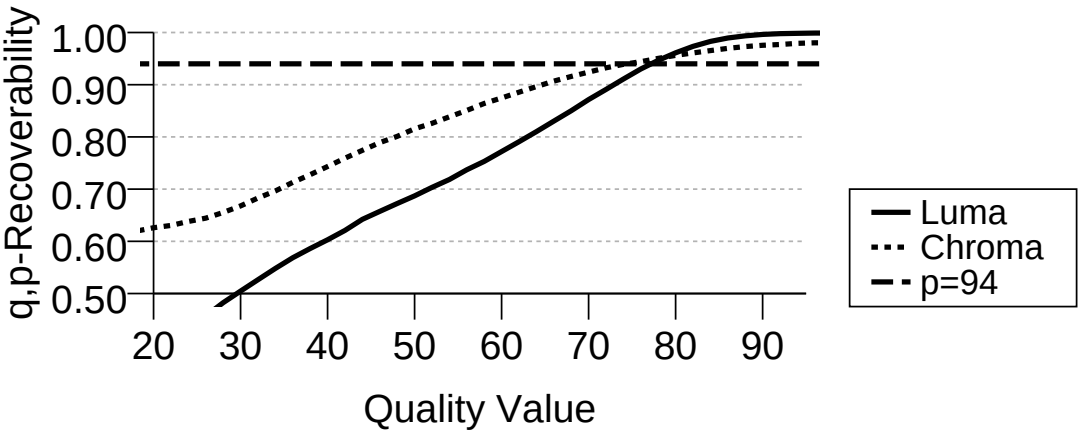

Figure 5.13: This indicates the feasibility of leveraging $RS(255, 223)$ to improve the q, p-Recoverability of a $\left(Y^3_{1\times1}, C^1_{2\times2}\right)$ protocol.

Comparison with Steganographic Efficiency. Though the goals of steganography and Cryptagram differ, both embed data in images, so we can compare the two in terms of bits/pixel efficiency.

Related work has expounded on the efficiency of steganographic embeddings [17, 95], reducing the approach to one embedding p message bits into $2^p - 1$ pixels, yielding a relative payload of $\alpha = p/(2^p - 1)$. While steganography choose slightly higher values of p a low value of p yields 0.42 bits per pixel for $p = 3$. In the highlighted row, our effective efficiency is 3.06 bits per pixel. In comparison, our approach represents a minimum 7.5× improvement.

5.7.5 File Format Embedding Comparison

In Figure 5.14, we show the q, p-Recoverability versus filesize ratio of JPEG versus webp image compression formats. By file ratio, we mean the on-disk size of the output image format for the same image canvas input. Notably, the embeddings are always three bits per pixel in the Figure. We see that for the same probability

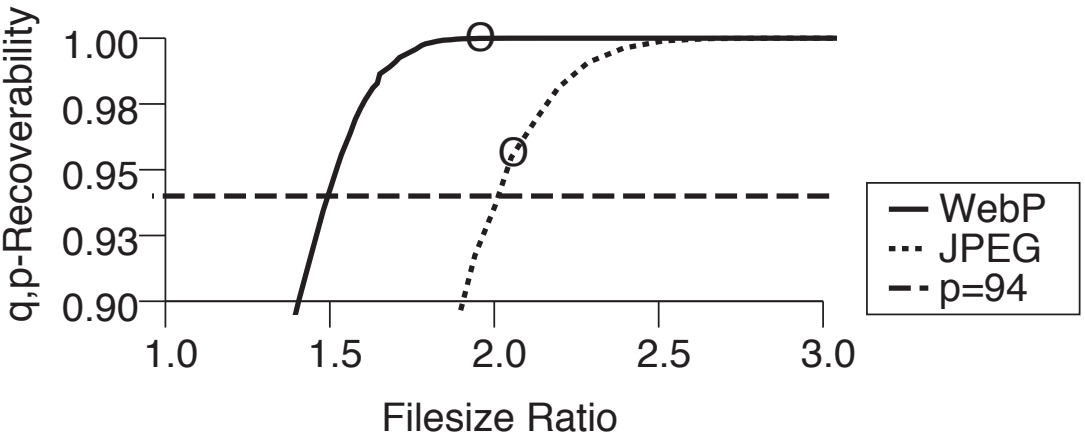

Figure 5.14: Showing the comparison of JPEG and lossy webp recoverability vs. filesize ratio. We draw the reader's attention to the $p = 94$ threshold as a point of comparison with the ECC studies in the rest of this paper. We acknowledge that JPEG and webp quality settings are not related and cannot be directly compared. However, this figure shows that for a similar notion of q, p-Recoverability, webp has a smaller filesize expansion than JPEG to achieve the same probability of recovery. To note the distinction in the meaning of "quality" between libjpeg and webp, we highlight the points along the curves where quality is 74 for each codec.

of recovery, p, webp has a much smaller filesize ratio than JPEG. As OSNs besides Google+ begin to experiment with webp deployment [71], the opportunity for lower bandwidth and storage requirements while maintaining q, p-Recoverability means that Cryptagram can be applied as improved media compression formats are adopted.

5.7.6 Deployment Usage Data

At the time of thesis publication, Cryptagram has nearly 400 active installations, with 373 users agreeing to participate in our IRB-approved study. Through this study, we receive high-level data about the Cryptagram encryption and decryption habits of our users. The following data does not include the authors'

own tests or images. We have had more than 3,300 Cryptagram image decryption events with more than 160 unique encrypted images generated. Of the decrypted images, we can confirm that 102 unique images have been decrypted from Facebook and 217 unique images from Google+.

5.8 Discussion

Applicability of q, p-Recoverability to Lossy Formats. OSNs continue to use lossy image formats in order to reduce demands on storage infrastructure and reduce delivery latencies to end-users. Recently developed formats should be considered given these goals. We have begun to examine the webp [33] format for Cryptagram. The tool of q, p-Recoverability applies in the analysis of these formats given that the spatial domain pixel value is the key component of Cryptagram communication.

Transformations. We aim to handle a variety of transformations with the development of q, p-Recoverable protocols. In previous sections, we discussed the design and evaluated our protocols' q, p-Recoverability with respect to the JPEG transformation. We have begun prototyping our approach to cropping and noising transformations on images produced by Cryptagram as well, leveraging blocking algorithms coupled with ECC. While we do not address rotation explicitly, we do not consider such a transformation intractable as we apply techniques from QR Codes (two dimensional barcodes) by orienting corner features in future iterations.

Scaling transformations are of interest given the pervasiveness of lower resolution images (e.g., thumbnails) to partially depict images on a social networking website. We have considered the integration of pyramid representations [12, 18] in

the design of future embedding protocols to meet this transformation request.

5.9 Related Work

P3 [61] examined the use of non-colluding services to store minimally-revealing cleartext images in one service and encrypted versions of DCT coefficients of JPEG images in another service. Their system experienced a 10-20% file size increase from the original compressed image when one follows their recommended privacy-preserving settings by setting the DCT-hiding threshold in the range $T \in [10, 20]$. The authors acknowledged their technique's vulnerability to face identification when $T \geq 35$. Cryptagram fundamentally differs from P3 in two ways. First, Cryptagram completely avoids the use of third parties. Secondly, Cryptagram works only in the encrypted bit space and does not expose any unencrypted data to the end-user. Unless users' keys are compromised, users cannot have their faces detected with any of our embedding protocols.

McAfee Social Protection lets users store cleartext photos on their server while uploading blurred versions to Facebook, then facilitates access requests [69]. This superficially addresses photo privacy, but in the end, amounts to an escrow service that redirects trust from one third party to another.

Steganography. Cryptagram is superficially reminiscent of various attempts to embed cryptographic data in JPEG through traditional steganographic techniques [28], but differs significantly from conventional JPEG steganography. Cryptagram makes obvious that it is hiding data to attain greater efficiency, and furthermore, does so in a way that is robust to image compression, which steganography generally is not. Attempts to achieve lossy compression tolerant steganography

are early works with inefficient results [40].

One recent work attempted to embed data in JPEG DCT coefficients without the steganographic constraint of being hidden. The non-linearities of DCT quantization and rounding in standard compression and decompression required very conservative data embedding that resulted in efficiency significantly lower than what we were able to achieve [4].

Li et al. [47] address the concern of hiding data within DCT coefficients but shuffle the DCT coefficients between blocks that then are quantized during the JPEG compression algorithm. Likewise, Poller et al. demonstrate that DCT coefficient permutation and spatial domain permutation do provide some security features but do not address efficiency or prove the correctness of their security mechanism [58].

A formalization for the description of the embeddings that we use in Cryptagram have been explored by Galand and Kabatiansky [29] but the authors do not explore how to construct such protocols.

But Cheddad et al. [16] claim that spatial domain techniques are not robust against noise, only work in bitmap-formatted images, and are not robust against lossy compression and image filters. Cryptagram overcomes all of these drawbacks in spatial domain embedding and demonstrates the useful privacy and security properties that can be available for OSN photo sharing. Cryptagram achieves q, p-Recoverability in the face of the complete recompression of the JPEG image containing sensitive information.

5.10 Conclusions

The advent of popular online social networking has resulted in the compromise of traditional notions of privacy, especially in visual media. In order to facilitate convenient and principled protection of photo privacy online, we have presented the design, implementation, and evaluation of Cryptagram, a system that efficiently and correctly protects users' photo privacy across popular OSNs. We have introduced q, p-Recoverability and demonstrated Cryptagram's ability to embed cryptographic primitives correctly to attain q, p-Recoverability through JPEG compression in our implementation.

Chapter 6

Conclusion

Privacy in the age of social media highlights a complex interplay of human values. A piece of technology can alter social interactions and, in turn, alter society itself. And while social media offers convenience it may threaten user privacy during a company's push to monetize or when system administration goes awry. This work represents different approaches for addressing visual privacy on the client side. The projects have the shared goal of changing the way visual information flows in an attempt to support contextual integrity.

While a top-down approach to visual privacy would create new social media from scratch for maximum flexibility, it would suffer from the challenge of *going viral*. In contrast,the approaches in this thesis are bottom-up *hacks*. Using a mix of software and human computing, we demonstrated the possibility of these technical interventions. The systems run independent of social media providers but nonetheless can be used to alter informational flows in several key scenarios. While extraction and obfuscation were used to alter the visual attributes of transmitted human forms, encryption was demonstrated to alter the recipients of visual infor-

mation. These technologies have the potential to modify human behavior and even affect change behind the walled garden of social media providers.

Bibliography

[1] http://www.snapchat.com.

[2] High-End VFX Studio Time Estimates for MatchMoving. Private communication.

[3] Agarwala, A., Hertzmann, A., Salesin, D. H., and Seitz, S. M. Keyframe-based tracking for rotoscoping and animation. In *ACM Transactions on Graphics (ToG)* (2004), vol. 23, ACM, pp. 584–591.

[4] Backes, J., Backes, M., Dürmuth, M., Gerling, S., and Lorenz, S. X-pire! - a digital expiration date for images in social networks. *CoRR abs/1112.2649* (2011).

[5] Bakos, Y., Marotta-Wurgler, F., and Trossen, D. Does anyone read the fine print? testing a law and economics approach to standard form contracts. In *Testing a Law and Economics Approach to Standard Form Contracts (October 6, 2009). CELS 2009 4th Annual Conference on Empirical Legal Studies Paper* (2009), pp. 09–40.

[6] Bishop, C. *Pattern recognition and machine learning*, vol. 4. springer New York, 2006.

[7] boyd, d. m., and Ellison, N. B. Social network sites: Definition, history, and scholarship. *Journal of Computer-Mediated Communication 13*, 1 (2007), 210–230.

[8] Boyle, M., Edwards, C., and Greenberg, S. The effects of filtered video on awareness and privacy. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work* (2000), ACM, pp. 1–10.

[9] Bregler, C., Bhat, K., Saltzman, J., and Allen, B. ILM's Multitrack: a new visual tracking framework for high-end VFX production. In *SIGGRAPH 2009: Talks* (2009), ACM, p. 29.

[10] Bregler, C., and Malik, J. Tracking people with twists and exponential maps. In *Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference on* (1998), IEEE, pp. 8–15.

[11] Buchanan, A., and Fitzgibbon, A. Interactive feature tracking using kd trees and dynamic programming. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2006), vol. 1.

[12] Burt, P. J. Fast filter transform for image processing. *Computer graphics and image processing 16*, 1 (1981), 20–51.

[13] C-Mon and Kypski. One frame of fame. `http://oneframeoffame.com/`.

[14] Canny, J. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 6 (1986), 679–698.

[15] Chaquet, J. M., Carmona, E. J., and Fernández-Caballero, A. A survey of video datasets for human action and activity recognition. *Computer Vision and Image Understanding* (2013).

[16] Cheddad, A., Condell, J., Curran, K., and Mc Kevitt, P. Digital image steganography: Survey and analysis of current methods. *Signal Processing 90*, 3 (2010), 727–752.

[17] Crandall, R. Some notes on steganography. *Posted on steganography mailing list* (1998).

[18] Crowley, J. L. A representation for visual information. Tech. Rep. CMU-RI-TR-82-07, Robotics Institute, Pittsburgh, PA, November 1981.

[19] Daniel, A. CLOC: Count Lines of Code. http://cloc.sourceforge.net/.

[20] Deutscher, J., Blake, A., and Reid, I. Articulated body motion capture by annealed particle filtering. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on* (2000), vol. 2, IEEE, pp. 126–133.

[21] Duce, D., and Boutell, T. Portable network graphics (png) specification. *Information technology ISO/IEC 15948* (2003), 2003.

[22] Duell, M. Mark Zuckerberg's private Facebook photos revealed: Security 'glitch' allows web expert to access billionaire's personal pictures. *The Daily Mail (MailOnline)* (December 2011). http://www.dailymail.co.uk/news/article-2070749/Facebook-security-glitch-reveals-Mark-Zuckerbergs-private-photos.html.

[23] Felzenszwalb, P., and Huttenlocher, D. Pictorial structures for object recognition. *International Journal of Computer Vision 61*, 1 (2005), 55–79.

[24] Finkenzeller, K. *Data Integrity*. Wiley Online Library, 2003.

[25] Fitts, P. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of experimental psychology 47*, 6 (1954), 381.

[26] Forsyth, D., Arikan, O., Ikemoto, L., O'Brien, J., and Ramanan, D. Computational Studies of Human Motion: Part 1, Tracking and Motion Synthesis. *Foundations and Trends in Computer Graphics and Vision*.

[27] Frey, B., and Dueck, D. Clustering by passing messages between data points. *science 315*, 5814 (2007), 972.

[28] Fridrich, J. *Steganography in Digital Media: Principles, Algorithms, and Applications*. Cambridge University Press, 2009.

[29] Galand, F., and Kabatiansky, G. Information hiding by coverings. In *Information Theory Workshop, 2003. Proceedings. 2003 IEEE* (2003), IEEE, pp. 151–154.

[30] Garfinkel, S. L., and Miller, R. C. Johnny 2: a user test of key continuity management with s/mime and outlook express. In *Proceedings of the 2005 symposium on Usable privacy and security* (2005), ACM, pp. 13–24.

[31] Good, J. How many photos have ever been taken? http://blog.1000memories.com/94-number-of-photos-ever-taken-digital-and-analog-in-shoebox.

[32] Google. Closure Tools. https://developers.google.com/closure/.

[33] Google. webp: A new image format for the Web. https://developers.google.com/speed/webp/.

[34] Horn, B. K., and Schunck, B. G. Determining optical flow. *Artificial intelligence 17*, 1 (1981), 185–203.

[35] Horton, J. The dot-guessing game: A fruit fly for human computation research.

[36] http://poser.smithmicro.com/poserpro.html. Poser Pro: Complete 3D Figure Design and Animation.

[37] http://vision.cs.uiuc.edu/annotation/. Vision At Large: large scale data collection for computer vision research.

[38] http://www.adobe.com/products/aftereffects/. Adobe After Effects.

[39] http://www.vicon.com/boujou/. Bojou Matchmoving Software by Vicon.

[40] Hwang, R.-J., Shih, T., Kao, C.-H., and Chang, T.-M. Lossy compression tolerant steganography. In *The Human Society and the Internet Internet-Related Socio-Economic Issues*, W. Kim, T.-W. Ling, Y.-J. Lee, and S.-S. Park, Eds., vol. 2105 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2001, pp. 427–435.

[41] Johnson, M., Egelman, S., and Bellovin, S. M. Facebook and privacy: it's complicated. In *SOUPS '12: Proceedings of the Eighth Symposium on Usable Privacy and Security* (2012).

[42] Khatib, F., Cooper, S., Tyka, M., Xu, K., Makedon, I., Popović, Z., Baker, D., and Players, F. Algorithm discovery by protein folding game players. *Proceedings of the Nat'l Academy of Sciences* (2011).

[43] Kipp, M. Anvil-a generic annotation tool for multimodal dialogue. In *Seventh European Conference on Speech Communication and Technology* (2001), Citeseer.

[44] Koblin, A. The sheep market. In *Proceeding of the seventh ACM conference on Creativity and cognition* (2009), ACM, pp. 451–452.

[45] Kosinski, R. J. A literature review on reaction time. *Clemson University 10* (2008).

[46] Lewis, J. Fast normalized cross-correlation. In *Vision Interface* (1995), vol. 10, Citeseer, pp. 120–123.

[47] Li, W., and Yu, N. A robust chaos-based image encryption scheme. In *Multimedia and Expo, 2009. ICME 2009. IEEE International Conference on* (2009), IEEE, pp. 1034–1037.

[48] Lin, K.-Y., and Lu, H.-P. Why people use social networking sites: An empirical study integrating network externalities and motivation theory. *Computers in Human Behavior 27*, 3 (2011), 1152–1161.

[49] Malone, T., Laubacher, R., and Dellarocas, C. Harnessing crowds: Mapping the genome of collective intelligence. *Report: CCI Working Paper 1* (2009).

[50] Microsoft Safety Security Center. Create strong passwords, 2013. http://www.microsoft.com/security/online-privacy/passwords-create.aspx.

[51] Milk, Chris. The Johnny Cash Project. `http://www.thejohnnycashproject.com/`.

[52] Mori, G., and Malik, J. Estimating human body configurations using shape context matching. *Computer VisionECCV 2002* (2002), 150–180.

[53] Murray, R. M., Li, Z., and Sastry, S. S. *Mathematical Introduction to Robotic Manipulation.* CRC Press, Baton Rouge, 1994.

[54] Neustaedter, C., Greenberg, S., and Boyle, M. Blur filtration fails to preserve privacy for home-based video conferencing. *ACM Transactions on Computer-Human Interaction (TOCHI) 13*, 1 (2006), 1–36.

[55] Nissenbaum, H. *Privacy in context: Technology, policy, and the integrity of social life.* Stanford Law Books, 2009.

[56] Parr, B. Facebook by the Numbers. http://mashable.com/2011/10/21/facebook-infographic/.

[57] Pfeil, U., Arjan, R., and Zaphiris, P. Age differences in online social networking–a study of user profiles and the social capital divide among teenagers and older users in myspace. *Computers in Human Behavior 25*, 3 (2009), 643–654.

[58] Poller, A., Steinebach, M., and Liu, H. Robust image obfuscation for privacy protection in web 2.0 applications. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series* (2012), vol. 8303, p. 1.

[59] Powell, J. *33 million people in the room: how to create, influence, and run a successful business with social networking*. Ft Press, 2008.

[60] Princeton University Office of Information Technology IT Security. Tips for creating strong, easy-to-remember passwords, 2013. http://www.princeton.edu/itsecurity/basics/passwords/.

[61] Ra, M.-R., Govindan, R., and Ortega, A. P3: Toward Privacy-Preserving Photo Sharing. In *Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation* (2013), USENIX Association.

[62] Ramanan, D., Forsyth, D., and Zisserman, A. Strike a pose: Tracking people by finding stylized poses. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on* (2005), vol. 1, IEEE, pp. 271–278.

[63] Rana, M., Taylor, G., Spiro, I., and Bregler, C. 3d skeletal reconstruction from low-resolution multi-view images. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on* (2012), IEEE, pp. 58–63.

[64] Roberts, G., Carter, S., Ward, J., Birdsall, P., Rittler, S., and C, B. Mariano rivera, king of the closers – interactive feature. *New York Times Magazine* (July 2010).

[65] Rose, R. *MacVisSTA: A System for Multimodal Analysis of Human Communication and Interaction*. PhD thesis, Citeseer, 2007.

[66] Russell, B., Torralba, A., Murphy, K., and Freeman, W. LabelMe: a database and web-based tool for image annotation. *International Journal of Computer Vision 77*, 1 (2008), 157–173.

[67] Russell, B., Torralba, A., Murphy, K., and Freeman, W. Labelme: a database and web-based tool for image annotation. *International journal of computer vision 77*, 1 (2008), 157–173.

[68] Saini, M., Atrey, P. K., Mehrotra, S., and Kankanhalli, M. Adaptive transformation for robust privacy protection in video surveillance. *Advances in Multimedia 2012* (2012), 4.

[69] Security, M. McAfee Social Protection. https://apps.facebook.com/socialprotection/.

[70] Sengupta, S., and OBrien, K. J. Facebook Can ID Faces, but Using Them Grows Tricky. *The New York Times* (2012).

[71] Shankland, S. Facebook tries Google's WebP image format. http://news.cnet.com/8301-1023_3-57580664-93/facebook-tries-googles-webp-image-format-users-squawk/.

[72] Shi, J., and Tomasi, C. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on* (1994), IEEE, pp. 593–600.

[73] Shirky, C. *Cognitive surplus: Creativity and generosity in a connected age*. ePenguin, 2010.

[74] Sidenbladh, H., Black, M., and Fleet, D. Stochastic tracking of 3d human figures using 2d image motion. *Computer VisionECCV 2000* (2000), 702–718.

[75] Sminchisescu, C., and Triggs, B. Covariance scaled sampling for monocular 3d body tracking. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on* (2001), vol. 1, IEEE, pp. I–447.

[76] Snow, R., O'Connor, B., Jurafsky, D., and Ng, A. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (2008), Association for Computational Linguistics, pp. 254–263.

[77] Sorokin, A., and Forsyth, D. Utility data annotation with Amazon Mechanical Turk. *Proc of First IEEE Workshop on Internet Vison at CVPR 2008*.

[78] Spiro, I. Motion chain: a webcam game for crowdsourcing gesture collection. In *Proceedings of the 2012 ACM annual conference extended abstracts on Human Factors in Computing Systems Extended Abstracts* (2012), ACM, pp. 1345–1350.

[79] Spiro, I., Huston, T., and Bregler, C. Markerless motion capture in the crowd. *arXiv preprint arXiv:1204.3596* (2012).

[80] Spiro, I., Taylor, G., Williams, G., and Bregler, C. Hands by hand: crowd-sourced motion tracking for gesture annotation. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference* (2010), IEEE, pp. 17–24.

[81] Stone, Z., Zickler, T., and Darrell, T. Toward large-scale face recognition using social network context. *Proceedings of the IEEE 98*, 8 (2010), 1408–1415.

[82] Surowiecki, J. *The Wisdom of Crowds*. Anchor Books, New York, 2004.

[83] Tapscott, D. *Grown Up Digital: How the Net Generation is Changing Your World HC*, 1 ed. Mcgraw-Hill, 2008.

[84] Taylor, G., Hinton, G., and Roweis, S. Modeling human motion using binary latent variables. In *Adv. in Neural Inf. Proc. Sys.* (2007), pp. 1345–1352.

[85] Taylor, G., Spiro, I., Bregler, C., and Fergus, R. Learning invariance through imitation. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on* (2011), IEEE, pp. 2729–2736.

[86] Ternovsky, A. http://www.chatroulette.com/.

[87] The Chromium Authors. libjpeg — The Chromium Projects. `http://src.chromium.org/viewvc/chrome/trunk/src/third_party/libjpeg/`.

[88] Tierney, M., Spiro, I., Bregler, C., and Subramanian, L. Cryptagram: Photo privacy for online social media. In *Proceeding of the first ACM conference on Online Social Networks* (2013), ACM.

[89] Torresani, L., Hertzmann, A., and Bregler, C. Nonrigid structure-from-motion: Estimating shape and motion with hierarchical priors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 30*, 5 (2008), 878–892.

[90] Union, I. T. Digital Compression and Coding of Continuous-tone Still images. CCITT Rec. T.81, 1992.

[91] van der Maaten, L., and Hinton, G. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research 9* (2008), 2579–2605.

[92] Von Ahn, L. Games with a purpose. *Computer 39*, 6 (2006), 92–94.

[93] Vondrick, C., Ramanan, D., and Patterson, D. Efficiently scaling up video annotation with crowdsourced marketplaces. *Computer Vision–ECCV 2010* (2010), 610–623.

[94] Wang, L., Hu, W., and Tan, T. Recent developments in human motion analysis. *Pattern recognition 36*, 3 (2003), 585–601.

[95] Westfeld, A., and Pfitzmann, A. High capacity despite better steganalysis (f5–a steganographic algorithm). In *Information Hiding, 4th International Workshop* (2001), vol. 2137, Pittsburgh, PA, pp. 289–302.

[96] Wheeler, D. SLOCCount. http://www.dwheeler.com/sloccount/.

[97] Williams, G., Bregler, C., Hackney, P., Rosenthal, S., McDowall, I., and Smolskiy, K. Body signature recognition. Tech. rep., TR-2008-915, New York University.

[98] Williams, G., Taylor, G., Smolskiy, K., and Bregler, C. Body motion analysis for multi-modal identity. In *Int. Conf. Pattern Recognition* (2010).

[99] Williams, G., Taylor, G., Smolskiy, K., and Bregler, C. Body motion analysis for multi-modal identity verification. In *Pattern Recognition (ICPR), 2010 20th International Conference on* (2010), IEEE, pp. 2198–2201.

[100] Yuen, J., Russell, B., Liu, C., and Torralba, A. Labelme video: building a video database with human annotations. *ICCV, Kyoto, Japan 2* (2009).

[101] Zhao, Q. A., and Stasko, J. T. Evaluating image filtering based techniques in media space applications. In *Proceedings of the 1998 ACM conference on Computer supported cooperative work* (1998), ACM, pp. 11–18.

[102] Zimmermann, P. R. *The official PGP user's guide*. MIT press, 1995.