OpenScan: A Fully Transparent Optical Scan Voting System

Kai Wang
UC San Diego

Eric Rescorla
Skype

Hovav Shacham
UC San Diego

Serge Belongie
UC San Diego

Abstract

Existing optical scan voting systems depend on the integrity of the scanner. If a compromised — or merely faulty — scanner reports incorrect results, there is no ready mechanism for detecting errors. While methods exist for ameliorating these risks, none of them are entirely satisfactory. We propose an alternative: a radically open system in which any observer can simultaneously and independently count the ballots for himself. Our approach, called OpenScan, combines digital video recordings of ballot sheet feeding with computer vision techniques to allow any observer with a video camera to obtain a series of ballot images that he can then process with ordinary optical scan counting software. Preliminary experimental results indicate that OpenScan produces accurate results at a manageable cost of around $1000 in hardware plus $0.0010 per ballot counted.

1 Introduction

Optically scanned paper ballots are widely regarded by computer scientists as the most secure voting technology [11, 24] in common use. However, while in theory paper ballots are verifiable the reality is quite different; the actual ballot scanning and tabulating is performed by machines and there is no straightforward way for voters to verify that those machines are performing correctly.

The current "gold standard" approach for verifying the ballot scanning process is to perform a manual audit: once the ballots are scanned and subtotals have been published, a random sample of ballot batches are selected, hand-counted, and compared to the subtotals. In a "risk-limiting" audit [1] additional batches are counted until either all the ballots have been hand counted or there is strong statistical evidence that a full hand count would not change the winner of the contest. Because the audit is performed publicly, educated observers can convince themselves that the original tabulation was correct (or more properly, that a hand count would not change the outcome.) Unfortunately, existing batch-based auditing techniques are highly inefficient: the most efficient technique, described by Stark, [21], can easily require counting over 10% of ballots to confirm even a very lopsided victory in an election of moderate size[1] at weak risk limits [20]. Single-ballot techniques [14, 6] have been proposed and are far more efficient, but suffer from a number of practical difficulties involving chain of custody and ballot selection (see, for instance, Sturton et al. [22].) An additional difficulty with risk-limiting auditing techniques is that they require significant statistical sophistication, so it is not obvious to the layman that even a successful audit confirms the reported result. (A trusted third party could verify that the statistical computation is carried out correctly, but not necessarily the randomness by which audited precincts are chosen [9, 7].)

An alternative approach, recently demonstrated by the Humboldt Election Transparency Project,[2] is to rescan all the ballots with an independent scanner, and then publish the scanned images. Third parties can independently process the images (either manually or via optical scanning software) and compare their results to the reported subtotals. Unfortunately, rescanning approaches do not provide a comparable level of third party verifiability to audits because outsiders must trust the rescanning process, which — even it is performed independently — cannot be directly verified [18]. Third party verifiability requires performing a manual audit process to compare the rescanned records against the paper ballot records, thus negating the transparency and efficiency gains that rescanning is supposed to provide.

We consider a different approach, one that allows for an arbitrary number of third parties to simultaneously independently scan and tabulate the ballots, thus providing full third-party verifiability with minimal manual intervention. In this third approach, election officials allow observers to set up their own video cameras and then briefly display each ballot so it can be recorded. The observers can then use computer vision software to scan and count the ballots and even produce ballot images suitable as input to traditional optical scan software.

The idea for such a camera-based scanning system was first proposed by Anwar Adi in 2006, though we were

[1] Say, 50,000 to 100,000 votes. The median number of votes cast for President in the 2008 general election in a California county is 71,779, based on data from `http://www.sos.ca.gov/elections/sov/2008_general/`.

[2] Online: `http://www.humtp.org/`. Last visited 16 April 2010.

unaware of Adi's proposal when developing our system (see Section 1.2 for discussion of related work). In this paper, we describe the OpenScan system, which implements and extends Adi's idea to provide a system that is fully automated, practical for use in real-world elections with low error rates, and that separates ballot image reconstruction from counting, thereby increasing election transparency beyond what was envisioned by Adi.

1.1 The OpenScan System

In a conventional optical scan system, only election officials are able to directly observe ballot contents. Even if election officials publish complete ballot images or *cast vote records* (CVRs), voters must trust that those images accurately reflect the ballots. The Humboldt ETP partially ameliorates this limitation by providing a single additional set of images that is quasi-independently generated (the ETP staff have a special relationship with the registrar of voters). Clearly, however, this approach cannot scale to an arbitrary number of observers. Each new scan requires passing every ballot through yet another optical scanner and potentially damages the ballots, as well as presenting serious ballot custody issues. What is needed is a mechanism for allowing multiple observers to scan the ballots *in parallel*. However, this is not practical with commodity optical scanners, which require physical contact with the ballots.

A fundamental insight, due to Anwar Adi (see Section 1.2), is that fast, accurate ballot scanning is possible *without using a conventional optical scanner*. High resolution digital video cameras are readily available, inexpensive, and capture images of sufficiently high quality to use for ballot interpretation. If election officials allow observers to capture digital video of cast ballots, then the observers can use computer vision software of their choice to process the video to scan and count the recorded ballots. Several observers can participate simultaneously, each using her own video camera. Although at most one of these cameras can observe the ballots from an ideal angle, we show that computer vision techniques can be used to account for perspective and other distortion in the observed ballots, transforming the captured images so that they are rectangular. The entire system can be automated, including isolating images of each individual ballot, transforming the images so that they are rectangular, and processing them for ballot markings, with the result being tabulatable CVRs. Our OpenScan system is a proof-of-concept of such a camera-based scanning system for use in US-style real-world elections. OpenScan produces rectified ballot images from a video stream and feeds these images to Mitch Trachtenberg's Ballot Browser software for interpretation.

Because the observers are able to run software and hardware of their choice, the need to trust third parties is significantly reduced. In particular, there is no need to trust equipment provided by election officials. While in the majority of cases, we would expect observers to use a third-party (e.g., open source) scanning and interpretation package, it is possible for a technically minded (and especially paranoid) observer to construct her own camera-based scanning system out of commodity components (cameras, computers, generic computer vision libraries, etc.). While it is theoretically possible that these components might come compromised from the factory, this seems extremely unlikely given that each observer could use different components and thus in order to succeed an attacker would need to compromise devices from a large number of manufacturers.

Figure 1 provides an abstract overview of the expected workflow. Some mechanical ballot transport system (a sheet feeder, belt, etc.) is used to carry the ballots one at a time in front of a set of video cameras. We emphasize that this is the only interdependency between election officials and observers: election officials provide the observing opportunity and observers bring their own equipment and simply mount it as directed by officials.

In Figure 1 we see two cameras, A and B, both focused on ballot number 2. Each camera independently records each ballot, passes the video through a computer vision system which determines the boundaries of the ballot, corrects for distortion introduced by off-angle camera positioning, and produces a series of images, each of which represents a single ballot. These images are then passed to a vote processing system that determines which votes have been cast in each contest. In principle, the vote processing system could be the same as those already in use for central count optical scan systems. Many such systems operate with off-the-shelf scanners and we would simply be replacing those scanners with our camera plus computer vision apparatus.

We envision three potential deployment scenarios for OpenScan:

> *Secondary scanning.* In the simplest and least intrusive scenario, OpenScan is deployed independently of the official tabulation. The ballots are passed in sequence through some sort of sheet feeding mechanism which briefly exposes each ballot. Observers can then record them and perform their own tabulation. This scenario, unlike the others, is compatible with both precinct count and central count optical scanning, since the recording process can be performed with ballots which were previously scanned in the precinct.
>
> *Parallel scanning.* If the optical scanner is constructed so that each ballot is at some point exposed

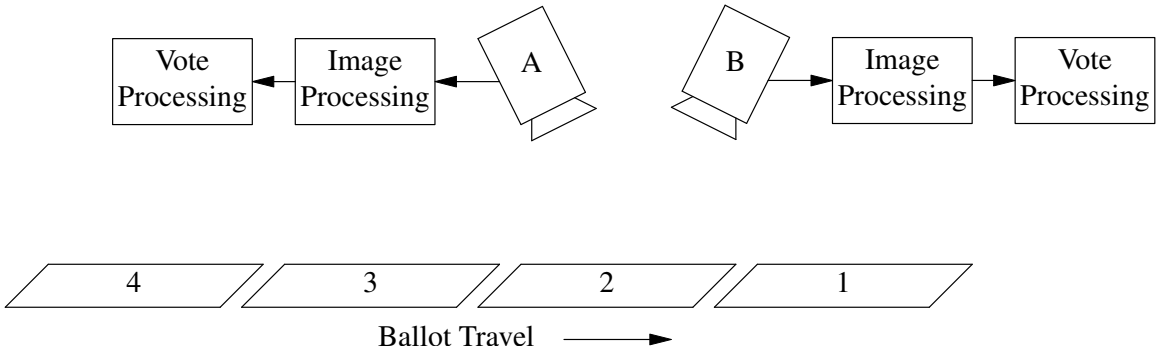


Figure 1: Overview of OpenScan.

(either before or after scanning), then the ballots can be recorded during the official scanning process. Thus, we in parallel construct an official record and an unofficial independent record. Indeed, the entire ballot need not be exposed at once: the computer vision system could reconstruct an entire ballot from several slices exposed over time (though our prototype does not support this mode of operation).

Primary scanning. Finally, we can imagine a system in which we discard the official optical scanner entirely and simply have a sheet feeder, with the official tabulation being done by an OpenScan-like system, in parallel with the independent tabulations. Clearly, this scenario places the highest demands on the speed and accuracy of the system, as it is being used for the official tally, not just as an independent check.

Although our (naïve) prototype implementation processes images much more slowly than our camera produces them, the SIFT algorithm at its core is amenable to optimization, parallelization [25, 10], and GPU implementation [19]. With an optimized implementation backed by GPU hardware, it should be possible to perform image and ballot processing as fast as the ballots are fed and scanned. This would allow real-time reconciliation between the multiple independent observers, each of whom would also have his system's raw recorded video stream for later analysis.

1.2 Related Work

OpenScan shows that it is feasible for independent election observers, using off-the-shelf equipment and automated computer vision techniques, to generate ballot images suitable for scanning and tabulation by optical scan software. It thus combines two lines of research: the first on producing ballot images for subsequent verification, the second on third-party optical scan at-a-distance using video cameras.

The first line of related work, on producing ballot images, is exemplified by the Humboldt Election Transparency Project, discussed above. In addition, Nagy et al. [16] describe a design for a precinct ballot counter based on a conventional still video camera mounted in a clear box. Their measurements give confidence that high accuracies are achievable under ideal conditions, but because their system only allows for a single camera (presumably controlled by voting officials) which is positioned directly in front of a single ballot and carefully illuminated, their system does not provide for any level of third party verifiability of the ballot contents.

The second line of related, on using video cameras and computer vision techniques to allow independent vote tabulation by election observers, was initiated by Anwar Adi, who in 2006 proposed the idea in the BlackBoxVoting.org forums [2]; Adi and his coauthors later presented a prototype implementation at VoComp 2007 [3].[3] We have also seen the idea of independent, manually triggered, cameras capturing single images independently suggested in private fora.

The fundamental contribution of our work compared to Adi's is that our use of SIFT and other more sophisticated computer vision techniques allows OpenScan to produce not vote counts but rectified ballot images suitable for posting (à la the Humboldt Election Transparency Project) or feeding into commodity ballot optical scan software for counting. In addition, compared to Adi et al.'s prototype implementation, OpenScan works with standard optical scan ballots used in US elections, rather than special-purpose ballots; and automatically recognizes ballots in a video stream, rather than requiring manual frame selection for each ballot. Finally, we provide in this paper a complete evaluation, including separate train and test cycles, showing that OpenScan can be used on real-world ballots with very low error rates.

The primary competitor to a system like OpenScan is end-to-end cryptographic voting; see, for instance,

[3] We are grateful to Doug Jones for bringing this line of work to our attention after our paper was submitted to EVT/WOTE.

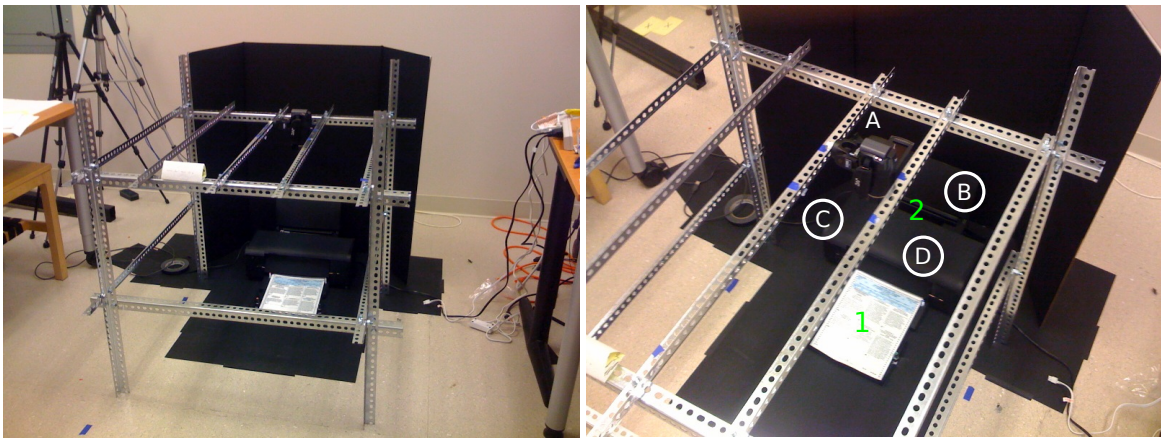


Figure 2: Our prototype system: scanning frame, camera, and printer-based sheet feeder. Letters A, B, C, and D represent the locations of the camera in our experiments. The labels 1 and 2 mark the location of the output tray and printer, respectively.

[8, 5]. Such systems provide third-party verifiable elections without any need to publish even cast vote records, let alone ballot images. By contrast, OpenScan makes full ballot images available to any observer who cares, and thus significantly reduces voter privacy; when a full ballot image is available, it is straightforward to mark ballots in a way that a third party can recognize. This appears to be a problem with any system which posts ballot images, and it is unclear how to alleviate it even with systems where the images can be processed before dissemination [18]. This is a case of a tradeoff between third-party verifiability and privacy.

The primary advantages of OpenScan vis-à-vis end-to-end systems is that it is conceptually far simpler: observers can immediately appreciate how OpenScan works even if they do not understand the details of the computer vision software. Moreover, no change at all is required in voter behavior — the only change required is by election officials who expose the ballots to recording.

2 System Description

2.1 Recording Process

Our prototype system is shown in Figure 2: Our sheet feeding mechanism (labeled 2) is a generic inkjet printer, an EPSON Workforce 30. Completed ballots are fed into the printer input paper tray and then we "print" blank pages to cause each sheet to advance through the printer and into the output tray. Each ballot thus appears in the output tray briefly face up before the next ballot is fed on top. The printer feeds at 30 pages/minute. At the base of the output tray, we added a small paper basket to catch and stabilize the ballots as they are printed.

The printer is surrounded by a rectangular steel frame which allows for stable camera mounting in a variety of positions. Our camera, a JVC Everio GZ-HM400, is then bolted to the frame pointing at the output tray. Because the position of the printer is fixed with respect to the camera, we can adjust the focus when the camera is mounted and then leave it fixed for the entire capture process. We capture video at 25 fps and at a resolution of 1920 x 1080, and record it on the onboard flash memory. At this recording speed, we can record up 2hr56m of video in the internal memory and another 2hr40min with an SDHC card. Our experiments (described in Section 3), used four ballot positions, corresponding to one quadrant of a square with three cameras on each side, so in principle at least 9 cameras could be mounted simultaneously while maintaining the same camera angles we are using.

2.2 Rectifying Ballot Images

Once we have recorded continuous video of the ballots, the next step is to process it into a series of rectified ballot images. The workflow is shown in Figure 3.

Adobe Premiere Elements The first step is to use Adobe Premiere Elements[4] to convert the full motion video into a series of discrete ballot images saved on disk as JPEG images. This is a relatively straightforward procedure using the 'Export As Sequence' option. Figure 4 shows sample frames from camera positions A and D.

Corner Detection The first stage in our pipeline is to run a Harris corner detector [12]. The primary purpose of this algorithm is to determine whether a frame has a low enough level of blur that we might plausibly be able to process it. If the number of corners returned by

[4] Online: `http://www.adobe.com/products/premiereel/`. Last visited 16 April 2010.

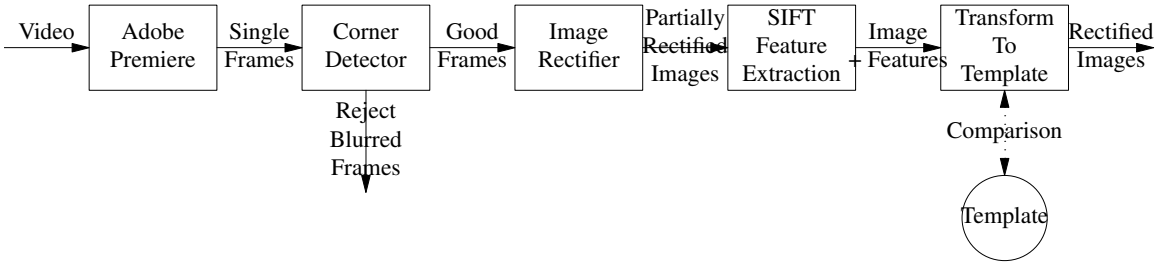
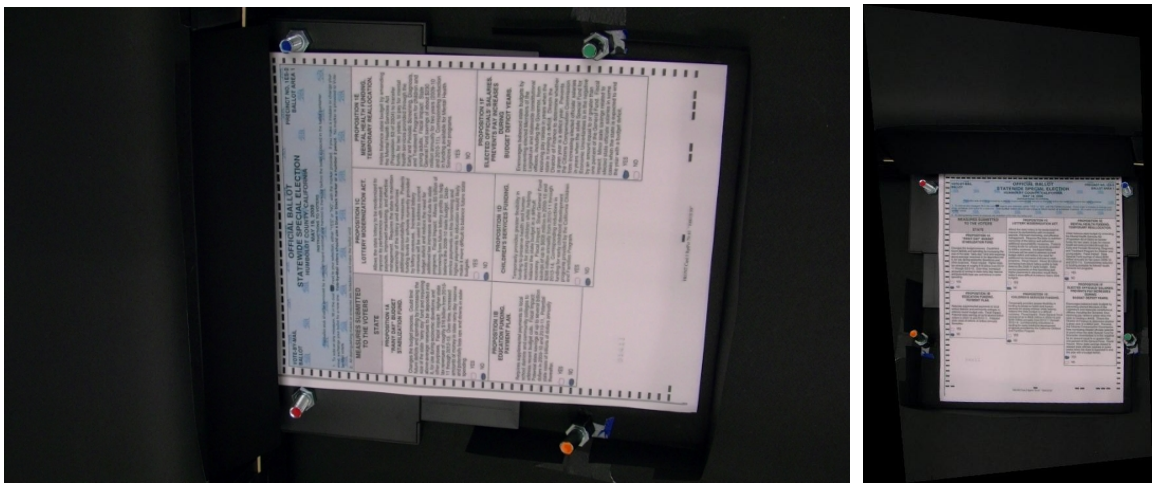


Figure 3: Processing stage 1: producing rectified ballot images.

(a) Position A

(b) Position D

Figure 4: Examples of video frames generated from positions A and D.



(a) Before calibration **(b)** After calibration

Figure 5: The video frame, before and after calibration.

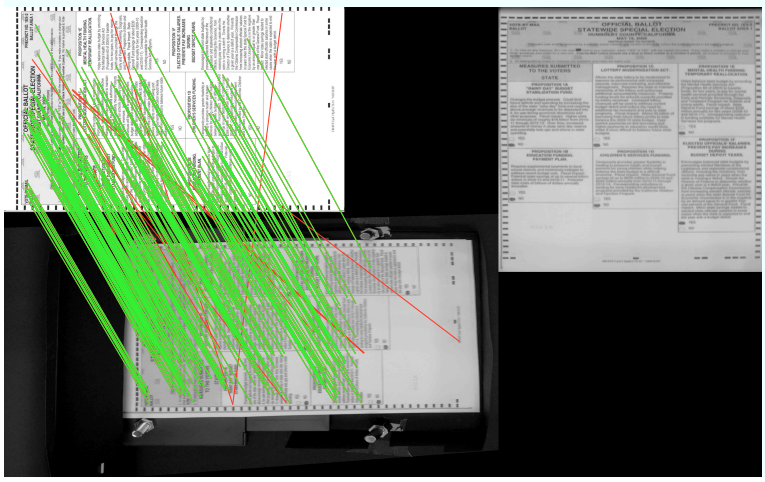


Figure 6: Feature extraction and matching. Green lines represent matches considered to be geometric inliers, while red lines are outliers. The right image shows the rectified video ballot.

the corner detector is below a threshold, we discard the frame. Otherwise, we pass the image to the next processing stage.

Initial Image Rectification Once overly blurred frames are discarded, we next perform an initial rectification pass. Prior to processing the first frame, we ask the user to calibrate the camera by clicking on four control points labeled in our rig. We use a single image of a ballot without any marks filled in as a template. Using these calibration points, we estimate a homography [13] between the video frame and template image to normalize the viewing angle. This provides an initial correction for camera angle — though this correction is incomplete because the ballots do not lie entirely flat in the output tray nor are they entirely straight. Figure 5 shows the frame before and after calibration.

SIFT The major image correction stage is performed by SIFT [15]. We use SIFT feature extraction and matching to find corresponding key points between the template and corrected video frame. After a correspondence is established, we estimate another — more precise — homography between the video frame and template, this time using the discovered keypoints. We then apply the homography to all keypoints and determine them to be either inliers — keypoints that are geometrically consistent with the transformation — or outliers. Figure 6 shows the the correspondences between keypoints in our template and a video frame.

2.3 Generating Unique Ballots

Once we have a set of fully rectified images, we then need to consolidate all the separate images of each ballot into a single image. This process, shown in Figure 7, occurs in two stages. First, we find all the images corresponding to the same ballot. Second, we merge all those images into a single image.

Identifying the Images of a Single Ballot Isolating the sequence of images of a single ballot uses two types of measurements: the locations of matching feature points and corner count. In order to detect the start of a sequence we check if a feature point matches in the upper 20% region of the ballot. This provides a strong cue that a new ballot has been produced (ballots are fed bottom first into the tray) and is sitting near the base of our tray. We consider this as the start of a new ballot sequence. We continue assigning ballots to the sequence until the corner count falls below a threshold, at which point we start ignoring frames until the start trigger is again observed.

The intuition behind the end trigger is that when a new page is entering the scene, the corner count falls precipitously due to the motion blur. Figure 9 shows the consistent pattern of corner count over time. Note that this corner detection method will not work well with a conveyor belt-style sheet feeder, since the amount of blur will be mostly constant. However, other mechanisms, such as looking for gaps between each page, would likely be usable to distinguish sheets.

Figure 8 shows this sequence of events. All frames occur between the start and end are collected and consolidated in the next step.

Although our prototype implementation assumes that all ballots are oriented the same way, it would be possible to handle different ballot orientations with essentially no overhead. SIFT features are invariant to rotation, so a

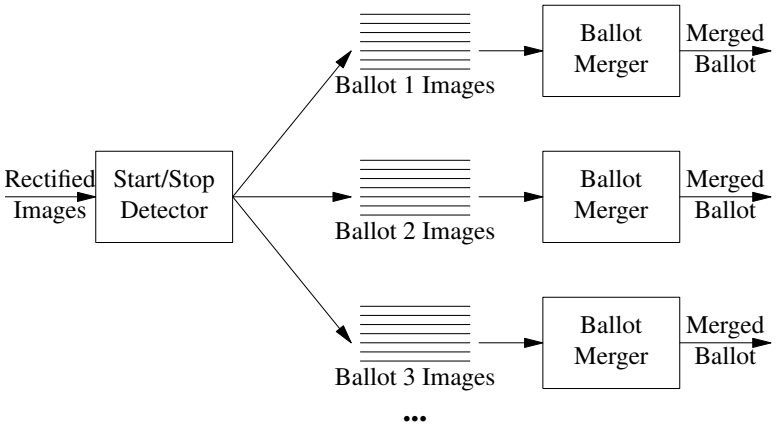


Figure 7: Processing stage 2: producing a single ballot image.

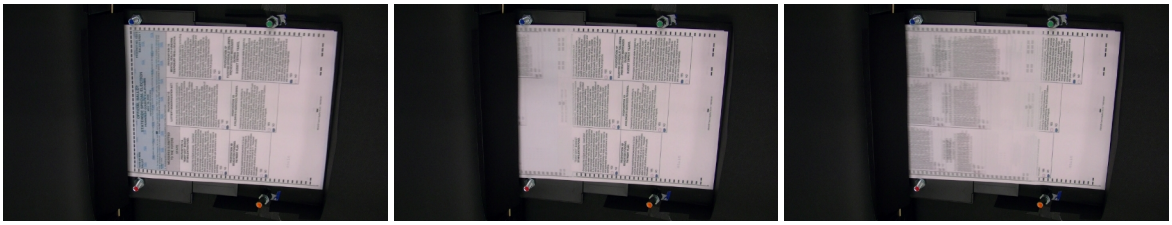


Figure 8: From left to right, frames corresponding to a start trigger, a middle frame, and an end trigger.

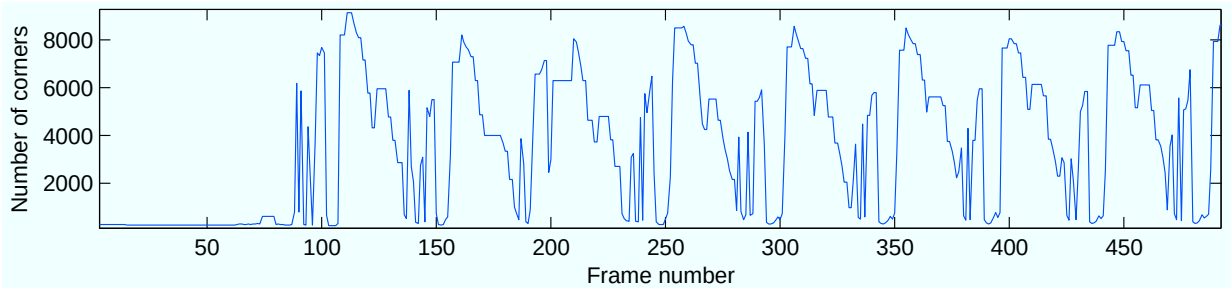
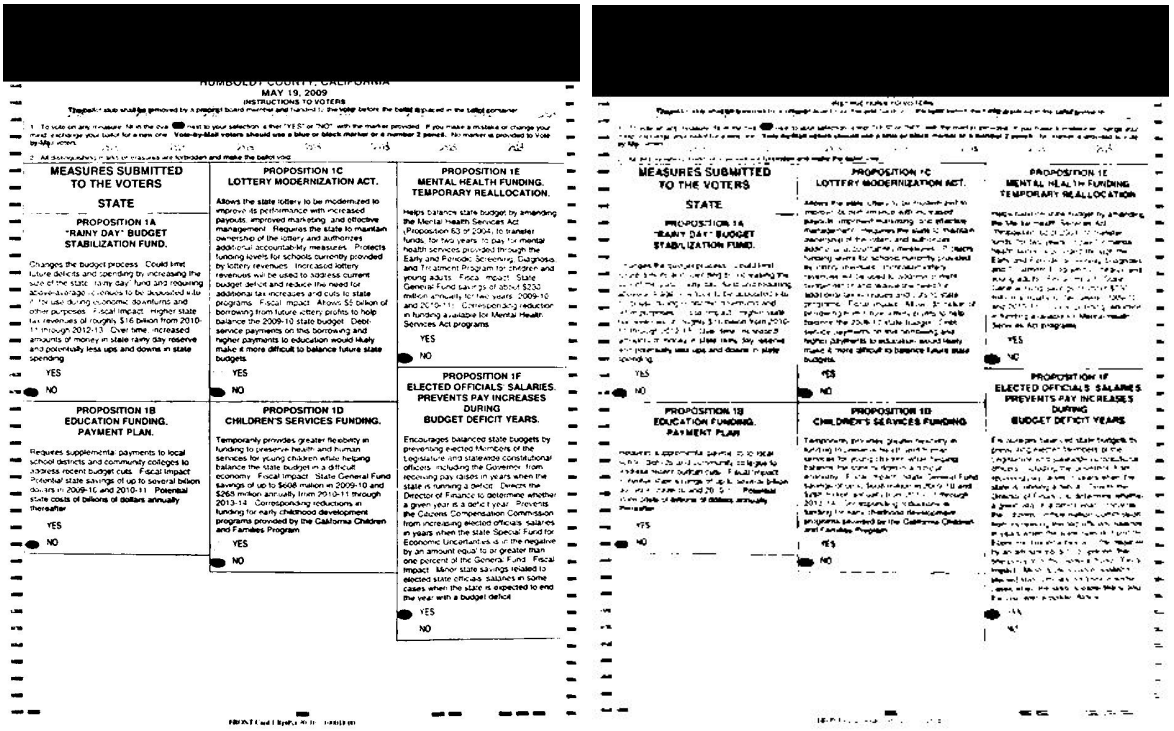


Figure 9: The number of corners detected over time, for the first 500 frames. We can see a repeating pattern as each ballot is produced, making it a powerful cue for distinguishing new ballots being output.



(a) **(b)**

Figure 10: High (a) and low-resolution (b) merged images of the same ballot.

single ballot template will match against any orientation; the estimated homography will capture and correct for the rotation. A small amount of additional logic could adjust the start trigger accordingly.

Producing a Consolidated Image The final computer vision stage is to consolidate all of the disparate images of a single ballot into a single image. In theory, these images should be identical but of of course in practice they are not. We merge the ballots on a pixel-by-pixel basis. First, we do black and white thresholding on each image, assigning each pixel of each image to be black or white. Then we give each image a weighted "vote" on every pixel as to whether it should be black or white, where the weight is determined by the number of matching feature points from that frame: more matching feature points means more influence on the final image.

More formally, let A represent an accumulation of votes indexed by pixel p as $A(p)$. Let B_i represent a ballot in a sequence of n associated ballots. Let M_i represent the number of matching feature points in frame i. We construct A as,

$$A(p) = \sum_{i=1}^{n} \begin{cases} M_i & : \quad B_i(p) = 1 \\ -M_i & : \quad \text{otherwise} \end{cases} \tag{1}$$

The final consolidated ballot F is,

$$F(p) = \begin{cases} 1 & : \quad A(p) > 0 \\ 0 & : \quad \text{otherwise} \end{cases} \tag{2}$$

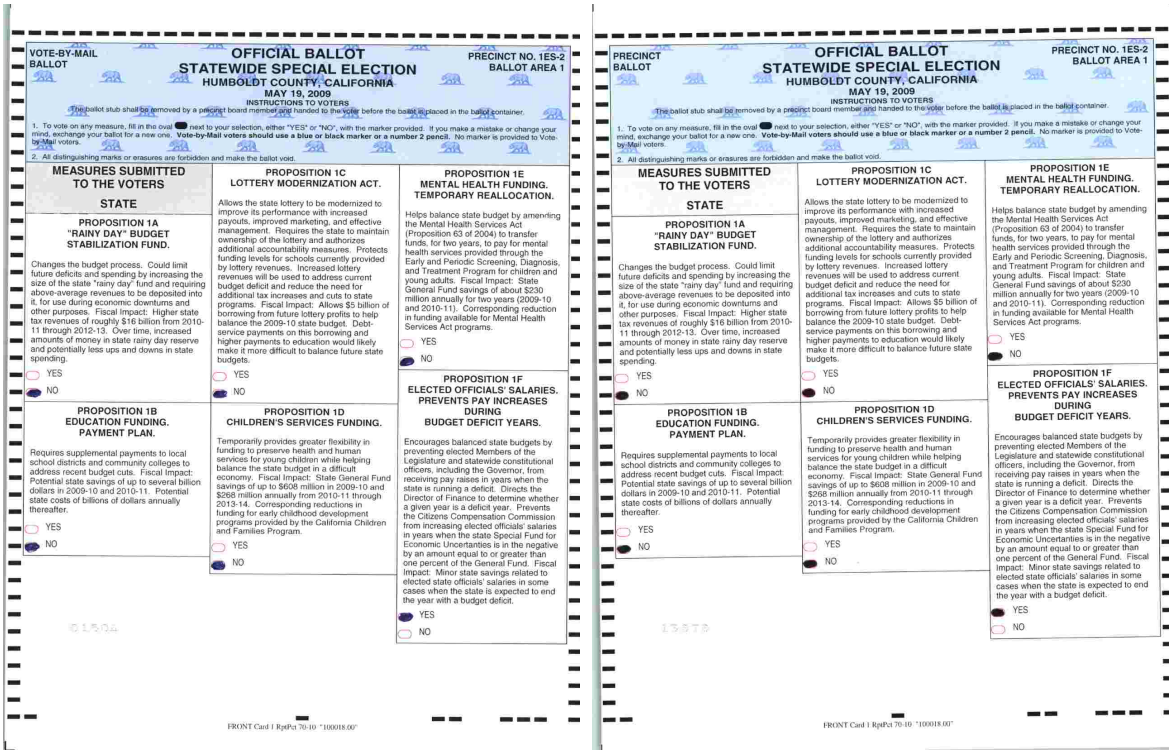
Figure 10 shows the output of this process. The left image is at full 952x1200 resolution, while the right has been resized to be only 600 pixels high. While the higher resolution image looks far better, the opscan targets in the smaller image are still quite visible.

The general problem of combining many low resolution images into a single high resolution image has been well-studied in computer vision and remains an active area of research. The work of [17] provides a detailed survey of the area. We use a simple pixel voting method in our approach, but plan to investigate more sophisticated methods of super-resolution in future work.

In our implementation, we used code from VLFeat[5] and Peter Kovesi's Computer Vision libraries.[6]

[5]Online: `http://www.vlfeat.org/`. Last visited 23 June 2010.

[6]Online: `http://www.csse.uwa.edu.au/~pk/Research/MatlabFns/`. Last visited 23 June 2010.



(a) (b)

Figure 11: The left image (a) is oriented in a way that is readable by Ballot Browser while the right (b) has very minor rotation and is unreadable. The sensitivity of the software motivated us to disable any internal pre-processing in our experiments.

2.4 Ballot Interpretation

At the end of the procedure in Section 2.3, we have a single black and white ballot image for each ballot. The final stage of the process is to determine what votes have been cast at each position. In our prototype we use the Ballot Browser software written by Mitch Trachtenberg for the Humboldt Election Transparency project. This software consumes images and outputs CVRs, so is suitable for our purposes. However, there is no inherent dependency on Ballot Browser; any optical scan processing software would be equally usable. We are using it purely for convenience.

3 Experimental Results

In this section, we describe our initial experiments to measure the accuracy of OpenScan.

3.1 Methodology

We began with a sample of 50 arbitrarily selected ballots downloaded from the Humboldt ETP from the May 19, 2009 special election.[7] These were single-sided ballots with 6 contests, each with 2 opscan targets. The images we have are 1275x1648 pixels and were subject to JPEG compression and thus display a number of visible compression artifacts. We printed each ballot on a sheet of ordinary printer paper and then fed those through our printer-based sheet feeder. We captured video from four camera positions, indicated by the letters A, B, C, and D in Figure 2, capturing the same 50 ballots from each position. These positions correspond to the viewing angles shown in Table 1.

Position	X angle	Y angle
A	0	0
B	20	0
C	0	15
D	20	15

Table 1: Camera angles in degrees.

We used a test/train protocol: we used an initial sample of 50 ballots to develop the system and tune parame-

[7] Online: `http://www.humtp.com/ballots.htm`. Last visited 23 June 2010.

ters. Once we were satisfied with the results, we used a separate batch of 50 ballots for testing. Those results are reported in the following section. In future, we would like to try with a much larger sample.

3.2 Results

In order to evaluate the results of OpenScan, we need to consider three metrics:

- Consistency between the results from multiple angles.
- Correctness with respect to manual interpretation of the ballots.
- Consistency with Ballot Browser's interpretation of the original images.

For all 50 ballots, our four camera angles agreed in all ballot positions (600 opscan targets). These results also match our manual interpretation of the ballots. Thus, for our limited trial we are achieving a 0% error rate. A larger sample would presumably allow us to estimate a more accurate, nonzero, error rate.

In our usage of Ballot Browser, we observed that ballot misalignment in the image was a significant source of error. When reading the original full resolution ballot images, Ballot Browser returns errors on 6 out of the 50 ballots (missing 72 marks), due to misalignments in the images (minor rotations of the ballots during the manual scanning process). Figure 11 shows an example of a readable and unreadable ballot by the original Ballot Browser. This sensitivity in the software prompted us to disable any image correction pre-processing performed by Ballot Browser in our experiments. In our approach, ballots from video are aligned automatically using SIFT to match the pose of a template, and thresholded to better manage lightly marked entries. In further experiments, we provided Ballot Browser with the original test ballot images aligned using SIFT and thresholded (in the same way as our video ballots) and observed it was able to read all markings with 0% error.

4 Discussion

In order to assess the suitability of OpenScan for the scenarios discussed in Section 1.1, we need to consider four major practical factors: accuracy, speed, cost, and compatibility with existing systems. This section treats each of these in turn.

4.1 Accuracy

While further experiments are required, the experiments described in Section 3 suggest that OpenScan is sufficiently accurate for independent auditing (the first two scenarios) of all but the closest contests. For comparison, Appel's precinct-level analysis [4] of the 2008 Minnesota Senate rate concludes that the "gross" error rate (the number of votes added or subtracted for a single precinct) of optical scanners was 99.91%. However, as this data only goes down to the precinct level, it excludes errors within a precinct which cancel each other out and so underestimates the per-ballot error.

The VVSG 2005 requires a very low error rate (one error in 500,000 ballot positions) [23], but the standards do not appear to require accurate results with imperfect ballots. Thus it is an open question whether a system like OpenScan has a high enough level of accuracy for primary scanning.

Note that even were it not possible to produce ballots sufficiently good to feed into ballot scanning software, just the ability to isolate individual ballots from video adds significant value: Ballot-based audits require the ability to pick a single identifiable ballot out of the entire corpus of cast ballots. Performing this selection manually presents a logistical barrier to deployment of such systems; a system like OpenScan can provide direct confirmation of the contents of a given ballot without having to go to the paper records. The relevant images can then be pulled out of the video stream and even if partial, one of the frames is likely to contain enough information to be comparable to the relevant CVR.

4.2 Performance

The major shortcoming of our current system is performance: on the Intel Core 2 (quad core, 2.83 GHz), it takes approximately 4 seconds to process a frame that passes the corner test. (Frames rejected earlier on are of course faster.) Around 25 frames per ballot pass the corner check. Thus, simply by processing frames simultaneously on the four CPU cores, we expect to be able to process all frames corresponding to a single ballot in about 25 seconds of wall clock time, or at about 1/12th of real time, given our current 30 pages/minute feed mechanism. It should be possible to obtain additional speedup in the SIFT algorithm at the heart of our frame processing by making use of the GPU [19]. Alternatively, we might improve performance by using vision algorithms specific to ballot images instead of the general-purpose SIFT algorithm. However, even with the existing software, it would be possible to process ballots as fast as they are scanned by using more CPU cores — most obviously by outsourcing processing to a cloud computing cluster like Amazon EC2, something we consider again in Section 4.3.

The more important question is whether we can maintain acceptable accuracy while improving speed: at 30

ballots/minute OpenScan is around an order of magnitude slower than high-end central count optical scan systems. For instance, the Sequoia Insight 400C can operate at up to 400 ballots/minute.[8] For primary and secondary scanning, this does not present an issue: OpenScan is so cheap that it is practical to simply purchase 10X more units (see Section 4.3).

However, for parallel scanning applications, we must match the rate at which the scanner operates, which would mean that we captured far fewer frames per ballot, with a potential loss of accuracy (our current camera would obtain less than 4 frames/ballot with the Insight 400C. Higher speed cameras are available and it may be possible to compensate in that way, but we have not made any serious investigation of this avenue.

4.3 Cost

The fixed cost of our OpenScan system is extremely low — a suitable camera is available for under $1000 and cheap printers are similarly inexpensive (and of course can be shared between users). Developing our prototype, including both writing the software and building the rig, took two weeks, full time, for a graduate student specializing in computer vision. An additional cost is computation, but as noted above, we can outsource that cost. A single core of our test machine can do approximately 15 frames/minute and is approximately the speed of a single EC2 compute unit.[9]

A single High-CPU Extra Large instance (8 virtual cores; 2.5 EC2 compute units per core) could process 300 frames/minute, or approximately 12 seconds of video per compute minute at 25 fps. With a feed mechanism of 30 ppm, this gives 6 ballots per minute. With approximately half the frames failing the corner test and requiring less processing, the effective rate is nearly doubled, to around 11 ballots per minute. At Amazon's current price of $0.68/hr/instance for this instance type, the cost would be approximately $0.0010/ballot, which is well within the range of even a relatively casual election observer: a million voter election would cost $1000 to fully recount via OpenScan. With our current software implementation, a cluster of three machines as powerful as EC2's High-CPU Extra Large instance would process the ballots in real time, i.e., as fast as our prototype paper-feed mechanism outputs them, around 30 ballots/minute.

Of course, central-count optical scanners, like other critical election headquarters devices, are not connected to the Internet, a practice for which there is ample good reason [11]. Official tabulation using OpenScan could therefore not make use of EC2 for OpenScan computation. Even so, third party observers could make use of EC2 — perhaps after the fact, if sufficient bandwidth is not available at election headquarters — since the security of their systems is not critical like that of the jurisdiction's official systems. Security flaws in these third-party systems exposed by connection to the Internet would allow attackers to obtain images of the scanned ballots, something they could obtain by setting up their own OpenScan system at election headquarters (but cf. [18]). Alternatively, it is possible to view our calculations above simply as estimates of the commodity CPU cost of deploying OpenScan.

4.4 Compatibility with Existing Systems

Aside from the performance issues mentioned in Section 4.2, there are a number of logistical challenges involved with using OpenScan in existing optical scan voting deployments. First, many jurisdictions use double-sided ballots. It is not entirely clear how to construct a version of OpenScan which will simultaneously scan both sides of the ballot. One possibility might be to pass the ballots over a clear plate so that cameras could be mounted above and below, but we do not have designs for this type of equipment. More likely, it would be necessary to make multiple passes through each ballot.

A second problem is that many existing central count optical systems allow ballots to be fed in in any orientation. A production version of OpenScan would need to adjust for this issue and straighten ballots regardless of their orientation. This is not likely to be especially difficult, however, as we simply need to determine the rough orientation in order to apply the right template. In the future, ballots could be marked to make this problem easier.

A final problem is ballot visibility. OpenScan requires that the entire ballot be visible to the camera (though it might be possible in principle to build a system in which the entire ballot was never visible at once). We have not investigated whether existing central count optical scan systems ever provide this level of visibility. If not, parallel scanning with OpenScan will be problematic. Note that this issue does not apply to either primary or secondary scanning applications, as those can use separate sheet feeding equipment.

4.5 Avenues for Future Work

While promising, OpenScan remains a prototype, and needs substantial additional engineering before it could be used even as an additional check via a secondary scan separate from the official tabulation.

First, we would like to have a completely integrated

[8] Online: `http://www.sequoiavote.com/documents/Insight400C.pdf`. Last visited 16 April 2010.

[9] Online: `http://aws.amazon.com/ec2/pricing/`. Last visited 16 April 2010.

system that could just be pointed at an (arbitrary) paper feed mechanism and would output a CVR for each ballot as soon as it was scanned. Such a system would be able to act as a check during the tabulation process. The current system is not yet fully integrated and so manual intervention is required to transfer images between Premiere and the ballot selection/rectification system and then again to Ballot Browser. This is a straightforward engineering task.

Second, OpenScan would need to be faster and more robust against various types of ballot reorientation. As discussed in Sections 1.1 and 2.2, we are using comparatively naïve algorithms so there is likely to be substantial room for improvement both in terms of speed and robustness. We would also need to add support for different ballot styles (this is mostly a matter of being able to support multiple templates).

Finally, a more thorough evaluation of accuracy is required. We would like to process a much larger ballot corpus and compare our results against those from a certified central count optical scanner in order to get a more accurate estimate of the accuracy of the system. This may require replacing Ballot Browser with a more robust ballot processing system. It may even be possible to plug our images into an existing ballot processing system that is designed to operate with a commodity scanner and thus is prepared to receive externally generated images.

5 Conclusions

OpenScan is a fully transparent optical scan counting system. OpenScan records digital video of voted ballots and uses computer vision techniques to produce a series of ballot images which can then be processed into Cast Vote Records. Because OpenScan works with inexpensive commodity digital video cameras, it allows multiple independent observers to simultaneously verify that optical scan ballots were correctly counted without requiring them to have any physical contact with the ballots, thus preserving the ballot chain of custody. Preliminary experiments suggest that OpenScan has an acceptable level of accuracy to be used for verification purposes; further work is required to determine if it is suitable for primary vote counting.

Acknowledgements

Thanks to Joseph Lorenzo Hall, Cullen Jennings, Don Molaro, and David Wagner for helpful conversations. Mitch Trachtenberg provided help in getting Ballot Browser working. Thanks also to our anonymous EVT/WOTE reviewers for their comments which greatly improved the paper, and Stephen Checkoway for his comments on the manuscript. We thank Brian Kantor and Kris Kitani for helpful discussions related to the construction of our prototype. Finally, as noted above, we are grateful to Doug Jones for bringing the related work of Adi to our attention after our paper was submitted to EVT/WOTE.

This material is based upon work supported by the National Science Foundation under Grant No. 0831532, CAREER Grant No. 0448615, and a Graduate Research Fellowship; and by a MURI grant administered by the Air Force Office of Scientific Research. Any opinions, findings, conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation or the Air Force Office of Scientific Research.

References

[1] Principles and best practices for Post-Election audits. ElectionAudits.org, Sept. 2008.

[2] A. Adi. Redundant counting design with voter verified paper ballot. Post thread in BlackBoxVoting.org forums. Online: `http://www.bbvforums.org/forums/messages/9707/17779.html`, Feb. 2006.

[3] A. Adi, W. Adi, M. Schüler, and P. Fröhlich. Demonstration of "open counting": A paper-assisted voting system with public omr-At-a-distance counting. In J. Benaloh, editor, *Proceedings of VoComp 2007*, July 2007. Online: `http://www.vocomp.org/papers/opencounting.pdf`.

[4] A. Appel. Optical-scan voting extremely accurate in minnesota. `http://www.freedom-to-tinker.com/blog/appel/optical-scan-voting-extremely-accurate-minnesota`, Jan. 2009.

[5] J. Benaloh. Simple Verifiable Elections, Aug. 2006.

[6] J. A. Calandrino, J. A. Halderman, and E. W. Felten. Machine-assisted election auditing. *USENIX/ACCURATE Electronic Voting Technology Workshop 2007*, Aug. 2007.

[7] J. A. Calandrino, J. A. Halderman, and E. W. Felten. In defense of pseudorandom sample selection. *USENIX/ACCURATE Electronic Voting Technology Workshop 2008*, July 2008.

[8] D. Chaum, R. Carback, J. Clark, A. Essex, S. Popuveniuc, R. L. Rivest, P. Y. Ryan, E. Shen,

and A. T. Sherman. Scantegrity II: End-to-End Verifiability for Optical Scan Election Systems using Invisible Ink Confirmation Codes, July 2008.

[9] A. Cordero, D. Wagner, and D. Dill. The role of dice in election audits—extended abstract. *IAVoSS Workshop on Trustworthy Elections 2006 (WOTE 2006)*, June 2006.

[10] H. Feng, E. Li, Y. Chen, and Y. Zhang. Parallelization and characterization of SIFT on multi-core systems. In *Workload Characterization, 2008. IISWC 2008. IEEE International Symposium on*, pages 14–23, Sept. 2008.

[11] J. A. Halderman, E. Rescorla, H. Shacham, and D. Wagner. You go to elections with the voting system you have: Stop-gap mitigations for deployed voting systems. In D. Dill and T. Kohno, editors, *Proceedings of EVT 2008*. USENIX/ACCURATE, July 2008.

[12] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference*, pages 147–151, 1988.

[13] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[14] K. C. Johnson. Election certification by statistical audit of voter-verified paper ballots, Oct. 2004.

[15] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, 2004.

[16] G. Nagy, B. Clifford, A. Berg, G. Saunders, D. Lopresti, and E. B. Smith. Camera-based ballot counter. In *10th International Conference on Document Analysis and Recognition*. IEEE, July 2009.

[17] L. C. Pickup. *Machine Learning in Multi-frame Image Super-resolution*. PhD thesis, University of Oxford, Feb. 2008.

[18] E. Rescorla. Understanding the security properties of ballot-based verification techniques. *2009 Electronic Voting Technology Workshop / Workshop on Trustworthy Elections*, Aug. 2009.

[19] S. N. Sinha, J. michael Frahm, M. Pollefeys, and Y. Genc. GPU-based video feature tracking and matching. Technical report, In Workshop on Edge Computing Using New Commodity Architectures, 2006.

[20] P. B. Stark. Efficient post-election audits of multiple contests: 2009 California tests. In *CELS 2009 4th Annual Conference on Empirical Legal Studies*, Los Angeles, California, Aug. 2009. University of Southern California.

[21] P. B. Stark. Risk-limiting post-election audits: P-values from common probability inequalities. *IEEE Transactions on Information Forensics and Security*, 4:1005–1014, Feb. 2009.

[22] C. Sturton, E. Rescorla, and D. Wagner. Weight, weight, don't tell me: Using scales to select ballots for auditing. *2009 Electronic Voting Technology Workshop / Workshop on Trustworthy Elections*, Aug. 2009.

[23] United States Election Assistance Commission. Voluntary voting system guidelines, 2005.

[24] Verified Voting Foundation. Resolution on electronic voting. Online: `http://www.verifiedvotingfoundation.org/article.php?id=5028`; retrieved 4/16/2010.

[25] Q. Zhang, Y. Chen, Y. Zhang, and Y. Xu. SIFT implementation and optimization for multi-core systems. In *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on*, pages 1–8, April 2008.