Big Brother: Recovering Hidden Writing from Pen Motion

Neil Patil
neilpatil@utexas.edu
UT Austin

Brian Cui
briancui@utexas.edu
UT Austin *

Abstract

Pen and paper is commonly used to store sensitive information. However, the widespread availability of digital cameras presents a growing threat to this medium: the ability to remotely record and analyze a victim's handwriting motions to determine what was written.

In this paper, we present Big Brother, an end-to-end program for recovering written digit sequences from pen motion recorded in video footage. Big Brother uses motion tracking and time sequence classification algorithms to compare a victim's pen motions against a library of pre-recorded example paths, thus discerning the written text even in cases where it is obscured. Big Brother can predict a set of guesses that dramatically outperforms brute-force random search, often identifying individual digits exactly. Given these results, we conclude that writing on pen-and-paper is not a secure storage mechanism in the presence of camera surveillance.

1 Introduction

Pen-and-paper is a widely trusted platform for the storage of sensitive information. From government forms to university exams, pen-and-paper is depended on as a secure "unhackable" record keeping medium. In recent years, high quality professional-grade cameras are becoming readily available for consumers - found in mobile phones, conference rooms, and and building security systems.

The growing prevalence of cameras forms a new threat against pen-and-paper: remote recording and offline analysis of handwriting footage to determine what a victim was writing.

As a victim draws characters on the page, their pen follows a trajectory dependent on the individual characters. A cursory evaluation of one's own handwriting may reveal an "O" projects a circular motion, whereas a "W" might appear more jagged. Analysis of this motion could allow an attacker to infer some properties of what was written.

Figure 1: A real-world scenario fitting our threat model. Even though the writing is not visible, the pen's motion can be observed. Big Brother generalizes this attack to less optimal camera positions.

We hypothesize that video footage of the pen's motion is sufficient to allow an adversary to reverse-engineer the original text written on the page. Such an ability would allow an attacker to estimate what the victim wrote, even if the victim had deliberately obscured their writing from view. Figure 1 shows an example of how a target subject could intentionally cover up their writing, but leave the body of the pen in view. Smartphone cameras, laptop webcams, and teleconference rooms present a wide surface area of internet connected cameras upon which an attacker can acquire the required footage.

To evaluate our hypothesis, we present Big Brother: the first ever system (to our knowledge) for recovering written digits from recorded video of handwriting. Our system assumes that an adversary has access to camera footage including a partial view of the victim's pen, but an indirect view of the writing surface. Our classifier estimates the path of the rear end of the pen using a general

*Both authors contributed equally.

purpose motion tracking algorithm, then compares it to pre-recorded three-dimensional sample sequences, transformed to match the camera's perspective. By combining a simple discretization heuristic with a temporal sequence similarity algorithm, Big Brother predicts a ranking of output predictions for each digit, spanning a set of guesses that would dramatically outperform a naive brute-force search against a sequence of the same length. In some cases, our system is able to predict the digit sequence exactly.

We focus on classifing number sequences to simplify the scope of our model. Digit sequences often constitute sensitive data: social security numbers, bank account numbers, and birthdays are commonly written personal information. Digit classification has also been studied rigurously in the field of optical character recognition (OCR) [15]. Although our task is more complex, the effectiveness of simple heuristics and feature engineering in OCR suggest machines are well suited for classifying handwritten numbers.

To evaluate the effectiveness of our model, we tested a suite of videos recorded from varying camera angles and distances. In our test footage, the victim writes with a font size equal to that used when entering social security numbers on the IRS W-9 form [1], emulating a real-world scenario.

We show that our system produces a ranking of digit outputs usable by an attacker to narrow the search space of possible digit sequences significantly. We theorize our model can easily extend to alphabetical characters, given a substantially larger training data set and improvements to our classification algorithm.

In summary, the contributions of this paper are:

- We investigate the feasibility of using video motion tracking to estimate pen behavior, and evaluate the effectiveness this data to classify handwritten digits.
- We demonstrate a novel information theft attack by building a system which can infer what a victim writes from camera footage of pen motion alone.
- We discuss several refinements that can be made to build a better classifier and further extend this class of attacks.

2 Background and Related Work

Side channel attacks spanning the physical space have increased in popularity in recent years, thanks to the growing availability of sensors in commodity consumer devices. Mobile keyloggers built from smartphone micrphones [3] and gyroscopes [5] demonstrate that hardware sensors can be exploited in unusual ways to extract signal from what might normally be considered environmental noise.

Our work was inspired by these novel examples of "noise analysis". In this paper, we seek to apply computer vision techniques to determine if *pen motion* can be observed and analyzed in a similar fashion.

Prior work involving pen observation has motivated some of our model design decisions. Using only a top-down camera view, Munich and Perona [11] constructed a pen-based human-computer interface similar to the stylus inputs available now on modern tablet PCs. The authors created this interface by using a combination of edge detection to track the pen tip and a Kalman filtering model to determine when the pen was pressed against the paper. Seok et al. extended this work in order to track pens against non-blank paper (which can include diagrams or printed text) using a color and shape matching strategy to detect the pen tip [14].

These models focus on tracing the front (writing) tip of the pen, but their effectiveness led us to believe that general-purpose motion tracking algorithms could be used to track the rear end of the pen as well. We further discuss our motion tracking methods in Section 4.4.

Wu et al. [16] augmented a pen by attaching a 3D printed dodecahedron with printed tracking marks on each face, enabling real time pen position tracking in six degrees of freedom with submilimeter accuracy. While our security model is meant to encompass "normal" commodity pens without predefined tracking markers, the effectiveness of the work by Wu et. al suggests that knowledge of the rear end of the pen can be used to infer motion of the front.

Yasuda et al. [17] proposed a signature verification system which compares the motion of a pen's tip against a pre-recorded signature path. In order to compensate for variations in signature size and signing speed, the authors applied a dynamic programming algorithm to compute the "warped" distance between pen paths. Our model takes advantage of a similar metric, Time Warp Edit Distance (TWED) [10], in order to normalize and compute the distance between temporal sequences. TWED is further discussed in Section 4.5.

3 Security Model

We assume that the attacker has access to a high resolution (1080p and above) camera capable of recording video of at least 30 frames per second (FPS), capabilities available on a broad range of mobile devices. Additionally, we assume that the camera has a clear view the rear end of the pen, but not necessarily the pen tip or the page itself. For example, the subject could be writing on the page and covering the text with their hand, but leaving the rear of the pen visible. An example frame capture

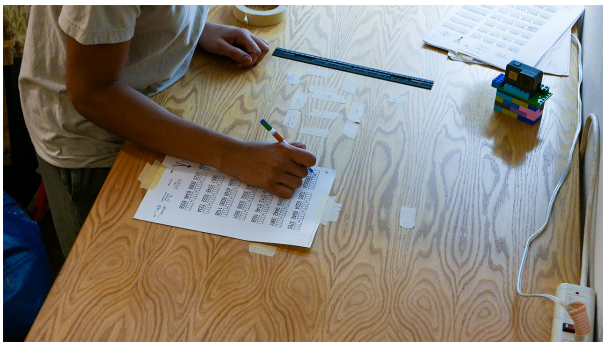

Figure 2: A frame capture of a test video fitting our security model.

from one of our test videos that demonstrates this view is shown in Figure 2.

Furthermore, we restrict the class of written symbols the attacker is attempting to decipher to the digits 0-9, and assume the quantity of digits written is known. Numbers keep our search space tractable and within the scope of this project, and regularly contain sensitive information of known lengths such as credit cards or social security numbers.

4 System Design

The model we present in this paper is an end-to-end classification program which accepts a recorded video of a victim writing in accordance with our security model, and produces a ranking of outputs representing predictions for each digit. The attacker must provide the video and specify the location of the paper, the location of the rear end of the pen, and the quantity of digits written. Once these details are provided, the program executes the following high level steps in sequence:

1. The program estimates the pose of the camera relative to the paper.
2. The set of ground truth digit sequences are transformed to match the camera's perspective.
3. The rear end of the pen in the video footage is motion tracked to determine a path.
4. The path is partitioned into digit chunks using a discretization heuristic.
5. Each digit chunk is compared against the set of transformed ground truth paths from step 2, forming a ranking of guesses for each digit by similarity.

In the following sections, we describe these steps in more detail.

Big Brother was developed using Python 3.6.5 and the OpenCV 3.4.3 computer vision library [2], and tested on a 64-bit Desktop PC running Ubuntu 18.04. The project source code, including all training and testing data, is available online at `https://github.com/patil215/bigbrother`.

4.1 Input

Our program requires the following inputs from the user:

- Video footage of the user writing, with the rear end of the pen visible.
- The Euler angle between the paper and the camera. This angle can be supplied manually or automatically estimated from a video frame by indicating the four corners of the paper.
- A bounding box drawn around the rear end of the pen to motion track.
- The quantity of digits to predict, n.

We theorize a more advanced system could automatically determine the camera angle, paper location, bounding box, and digit quantity. We leave discussion of possible future optimizations in Section 7.

4.2 Pose Estimation

Pose estimation is the challenge of determining the position of a 3D object in a 3D world, from the the 2D view seen by the camera. Our model estimates the pose of the *camera* relative to the paper, a flat 2D surface in a 3D world. Because our system is designed to work with the camera placed at an arbitrary angle, estimating the camera's pose is necessary before similarity metrics can be computed.

Mathematically, pose estimation is formulated as computing a rotation matrix R and translation vector t such that for a point w given in homogeneous world coordinates and a point p in camera coordinates,

$$p = \left[\begin{array}{c|c} R & t \end{array} \right] w$$

In expanded form, this looks like

$$sp = \left[\begin{array}{ccc|c} r_{00} & r_{01} & r_{02} & t_x \\ r_{10} & r_{11} & r_{12} & t_y \\ r_{20} & r_{21} & r_{22} & t_z \end{array} \right] w$$

where s is a scale factor depending on camera distance.

Estimating pose of a 3D object in the world can then be formulated as solving this equation for a set of *representative points* for which the 3D location on the model are known, and their corresponding 2D projections on the

camera viewing plane. Perspective-n-Point by Li et al. [8] is an effective optimization-based method for solving this task. We use the built-in implementation defined in OpenCV.

For our case, we use the four corners of an 8.5x11 inch page as representative points, giving the set $\{0,0,0\},\{8.5,0,0\},\{8.5,11,0\},\{0,11,0\}$, representing the top left, top right, bottom right, and bottom left corners respectively. When automatic angle detection is used, the user is required to indicate the four corners of the paper in the video.

4.3 3D Data Projection

Once the camera's angle is known, our model applies *linear transformations* to ground truth data so it can be viewed as if it was seen through the camera.

Using the rotation matrix R (or alternatively, three Euler angles corresponding to rotations along the X, Y, and Z axes) given in the previous step, we transform each point in the ground truth sequence by multiplying its 3D position by R. This step effectively transforms the ground truth motion sequences to what they would look like when viewed at the given camera angle, allowing us to compare a motion tracked test path against our ground truth set.

4.4 Motion Tracking

The task of visual object detection and tracking is an important computer vision challenge that has been studied for over two decades [18]. Effective motion tracking algorithms must continually estimate the location of objects in a 3D world projected on a 2D image, which can be visually occluded, disrupted by noise or artifacts, and quickly change location from one frame to the next.

In order to robustly track the motion of the pen, our model uses the Discriminative Correlation Filter with Channel and Spatial Reliability (CSRT) [9] [6] algorithm, a state-of-the-art motion tracking algorithm by Lukezic et al. that dynamically adjusts the object search region between frames based on optical flow, resulting in better performance with tracking non-rectangular objects.

We use the implementation of CSRT supplied by OpenCV 3.4.3. The program requires the user to draw a bounding box around the rear end of the pen as shown in the initial video frame. The box position and dimensions are then tracked through successive frames by the CSRT algorithm. Our model records the center point of the bounding box and timestamp for each video frame to construct a time series which estimates the pen's path in 2D. Finally, centering and normalization is applied to account for variations of scale in the data. An example motion tracked writing path of the digit "3" taken from our ground truth set is shown in Figure 3.

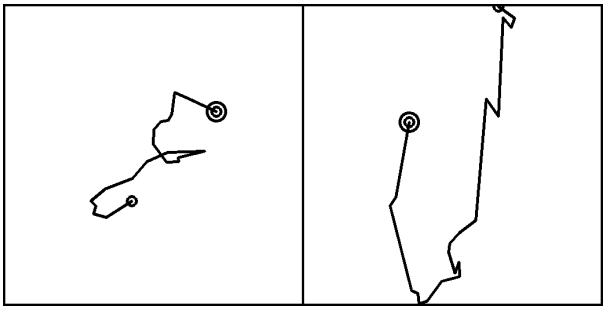

Figure 3: An example ground truth path for the digit "3", captured from motion tracking the rear end of the pen. The left image is from the top down XY perspective; the right image is from the side YZ perspective. The start and end of the path is designated by the larger and smaller circles, respectively.

A significant drawback to 2D motion tracking is a lack of depth estimation. Strategies for tracking the location of 3D objects in 3D space offer depth [7], but require a predefined 3D reference model for pose estimation. Instead, we included depth in our reference training data by using two cameras positioned orthogonally, which is then transformed as described above. Further details on how we recorded paths in 3D is explained in Section 5.

4.5 Time Sequence Similarity

Once a target pen path has been motion tracked, our algorithm compares it against the set of pre-recorded and transformed ground truth data. In order to compare two different temporal sequences, we define a metric representing a notion of "distance" between two sequences.

Prior work on quantifying temporal sequence difference has resulted in the Fast Dynamic Time Warping algorithm [13], which is commonly used due to its efficiency and time-invariance. Time Warp Edit Distance (TWED) [10] improves on this metric by making use of a "stiffness" hyperparameter, which adjusts the time elasticity of the sequence.

We make use of TWED in our algorithm, tuning the "stiffness" parameter to 0.0003 and delete cost parameter to 0.95, which optimizes our model against our single-digit test set. Because the stiffness penalty is most accurately applied when the time series for both sequences are similarly spaced, we interpolate both sequences to 25 evenly spaced points along the time scale prior to applying TWED.

The score of a sequence is then computed by taking the sum of squared distance for the individual X and Y axis temporal sequences. Thus, the score of a ground

truth sequence g when compared against a test sequence t is

$$dist(t,g) = TWED(t_x, g_x)^2 + TWED(t_y, g_y)^2 + P$$

where P is a time duration penalty, wihch we discuss in the next section.

As lower distances are better, the system attempts to find g which minimizes $dist$ for each t.

4.5.1 Time Duration Heuristic

The draw duration each digit can hint at the class of the digit. For example, a "1", which can be drawn with a single straight motion, is typically faster to draw than a "5".

Big Brother takes advantage of this characteristic by computing a range of acceptable time lengths for each digit class, according to the mean and standard deviation of samples in the ground truth set. If a digit's duration falls outside of this range, a penalty is added based on the magnitude of difference. This way, the classifier is more likely to guess digit classes with similar time durations.

4.6 Classification Algorithm

In order to classify digits using the distance score described in Section 4.5, we use a simplified version of a nearest neighbor classifier. In this formulation, our classifier simply ranks each of the ten digit classes by the minimum distance scores computed among the ground truth paths for each class. This is effectively a nearest-neighbor classifier with the neighbor count K set to 1.

The final output of the model is a ten class ranking for every digit in the sequence, with the highest ranking class for each digit the most likely digit output according to our algorithm.

In the following subsections, we describe our discretization strategy for turning a path composed of several digits into individual digit chunks for classification.

4.6.1 Discerning Spaces

Classification a sequence of digits requires distinguishing "pen down" motion (when ink is flowing) from "pen up" motion (spaces and transitions). To determine the temporal boundaries of each digit in the sequence, we make use of a simple space-detection heuristic. Spaces are almost always signified by short bursts of rapid motion, observed when the victim is shifting their hand in preparation for writing the next digit.

Our model applies the following iterative algorithm to find spaces: greedily find the point in the path with the largest speed (and is not already flagged as a space). Flag the nearby surrounding region as a space. Repeat until $n-1$ spaces have been found, where n is the number of digits to classify. The digit regions should then appear between and around spaces.

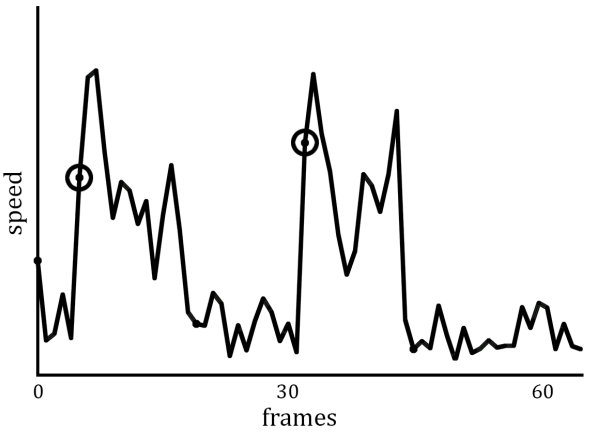

Figure 4: Chart showing normalized speed of the pen end for each frame in the sequence "1-2-3". The circles indicate the frame right after the beginning of spaces, as determined by our greedy heuristic.

While simple, this heuristic has proven in practice to be a reliable indicator of where spaces begin. Figure 4 displays the normalized speed of a path "1-2-3" from our ground truth set, with circles indicating the beginning of spaces according to our heuristic. We suspect more sophisticated heuristics can be used to improve our margin of error, which we discuss in Section 7.

4.6.2 Ranking Digits

While the start of the space is identifiable with a simple heuristic, we were unable to identify an effective heuristic for determining the *end* of a space. Thus, while the end of each digit (the beginning of each space) is well known, the beginning of each digit path chunk is not.

Our algorithm attempts to determine the correct starting frame of the digit iteratively. It does so by considering all digit start locations that correspond to a valid duration between the start and known end of the digit. Each chunk is then given a top ten ranking according to the distance metric and nearest-neighbor classifier described earlier. The starting point with the minimum score among these rankings is then used as the estimated start of the digit.

In our analysis, we have found this iterative approach capable of finding local minima of score as a function of starting frame, which are roughly correlated with the true starting location of each digit.

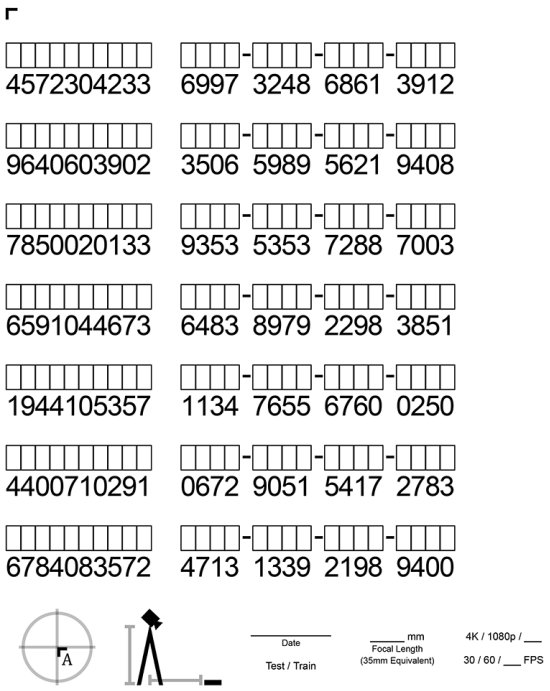

Figure 5: Example test bed template sheet, printed out on standard 8.5 in by 11 in paper. The digit entry boxes are .5 cm wide and .85 cm tall to emulate the digit entry fields on the IRS W-9 form [1]. The numbers are randomly generated with each test sheet.

5 Data Collection

All of our pen motion data was captured from a single test subject (Patil) writing with a nondescript office supply pen. Digits were written into squares measuring .5 cm wide and .85 cm tall, meant to mimic the number boxes on the IRS W-9 form [1]. An example of our test bed template is shown in Figure 5.

Due to time and labor constraints, we limited collection of example handwritten sequences to that of a single individual. Admittedly, this seriously limits how well we can infer how the model would generalize to different styles of handwriting. We further discuss the limitations to Big Brother's data collection strategy in Section 7.

5.1 Ground Truth

Our ground truth digit paths were collected using a two camera setup to capture the motion of the pen in 3D. Our camera hardware includes a professional 12MP Panasonic Lumix DMC-Lx100 camera and a 12MP GoPro Hero 5. We positioned the Panasonic Lumix camera di-

Figure 6: The camera arrangement used for recording training data. The Panasonic Lumix camera captures the top-down X-Y view of the template sheet, while the GoPro captures the horizontal Y-Z view of the pen motion.

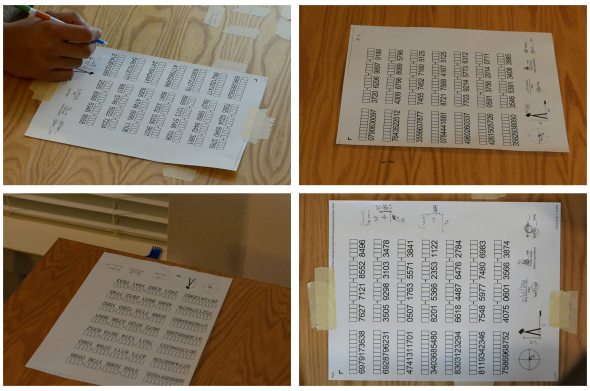

Figure 7: Frame captures of the (0, 55, 100), (0, -45, -90), (0, -30, -130) test cases and ground truth collection case respectively. The images here have been cropped, brightened and magnified for viewing clarity.

rectly over our test bed with a top down view of the paper, to measure the X and Y axes. We positioned the GoPro camera horizontally on the surface of the table, left of the test bed, to measure the Z axis (up and down) motion of the pen. Both cameras were set to record at 1080p and 60 frames per second (FPS). Figure 6 shows our real world camera arrangement.

All of our ground truth data consists of single-digit 3D path recordings of our test subject (Patil). Digits were drawn and recorded by both cameras simulataneously. The pen motion was motion tracked independently on footage from each camera and then normalized based on the real-world measured field of view for each camera. Finally, the paths were correlated by frame count (timestamp) to produce a 3D path. This path was then manually discretized to determine the time boundaries for each recorded digit. A sample 3D path is shown in Figure 3.

In total, our ground truth data included 253 single digit 3D paths, roughly evenly distributed among the ten possible digits.

5.2 Test Data

To evaluate the effectiveness of Big Brother, we recorded a suite of test videos comprising multiple camera angles, distances, and digit sequence lengths of our test subject (Patil) writing random numbers on the same test bed template shown in Figure 5. All videos were recorded with the Panasonic Lumix camera at 4K resolution and 30 FPS. Test paths were captured using the motion tracking algorithm as described in Section 4.4 after manually drawing a bounding box around the rear end of the pen. The following table provides a snapshot of the quantity of test cases, organized by sequence length.

Euler Angles	Distance (m)	1	4	7	9
0, 55, 100	2.4	80	56	32	16
0, -30, -130	1.4	69	49	28	14
0, -45, -90	1.7	70	49	28	14

Frame captures from each angle are shown in Figure 7. All in all, our test video data comprises 1213 paths spanning 12.4 gigabytes of compressed video footage. In this paper we focus on a representative subset of test paths of "interesting" length: credit card numbers are groupd by four digit sequences, phone numbers are seven digits, and social security numbers are nine digits. All other test sequences and results are available in our project's source code.

6 Evaluation

As longer digit sequences are exponentially more difficult for our model to classify, in our evaluation we discuss the likelihood of our model *considering* the correct sequence after allowing it to make b guesses for each digit.

We define the *guess magnitude* as the size of the set of possible digit sequences that may have been written on the page. The *baseline guess magnitude*, is the size of possible digit sequences provided by *random guessing*. Given $1 \leq r \leq 10$ guesses for each of the n digits in the sequence, the total number of sequences spanned by the guesses is r^n. In the worst case, $r = 10$ and the guess magnitude is 10^n, equivalent to brute force search.

Our model offers a ranking of all ten possible digits for each of the n digits in the sequence. When we consider only digits ranked r or better, the model's guess magnitude is r^n. We compare our model to a baseline random guess where each digit is randomly picked out of a size of set r. The model's *accuracy score* is computed as the likelihood that the correct digit sequence appears among all guesses encompassed by the guess magnitude. Given 10^n total n-digit sequences, the baseline random guess accuracy score is then $\frac{r^n}{10^n} = (\frac{r}{10})^n$.

Table 1 includes a breakdown of how well our model classifies digit sequences of length 1, 4, 7, and 9, and compares the model's accuracy score to the baseline. Overall, Big Brother performs consistently better than the baseline by a significant margin, given the same number of guesses (maximum rank considered), but scores decrease dramatically as n (and the search space) increases. Graphs comparing Big Brother's accuracy to the baseline for $n = 1, 4, 7$ are shown in Figure 8.

Interestingly, Big Brother's performance varies slightly depending on the positioning of the camera. For the single-digit case the average rank at the best angle (0, -45, -90) is 1.5, while at the worst angle (0, 55, 100), the average max rank is 2.0. This confirms that certain camera vantage points are more useful than others. Table 2 provides complete statistics of average max rank by sequence length and camera angle.

6.1 Single Digit Accuracy

Out of a test set of 219 single digit path samples, Big Brother scores considerably better than the baseline and is able to classify digits with 61.64% accuracy with only a single guess. Given a set of five guesses (50% probability for our baseline, i.e. flipping a coin), Big Brother manages to contain the correct digit within the set 95.89% of the time.

Perhaps the most significant factor contributing to Big Brother's success in the single digit case is the lack of dependence on surrounding digits and spaces for estimating digit chunk boundaries. The need to identify start and end locations for digits using our greedy space heuristic described previously in Section 4.6.1 greatly degrades performance in multiple digit sequence classi-

Sequence Length	1		4		7		9	
Max Rank	BB (%)	Base (%)	BB (%)	Base (%)	BB (%)	Base (%)	BB (%)	Base (%)
1	61.64	10.00	3.29	0.01	0.00	0.00	0.00	0.00
2	83.11	20.00	15.79	0.16	0.00	0.00	0.00	0.00
3	89.95	30.00	32.24	0.81	7.14	0.02	2.44	0.00
4	93.61	40.00	48.03	2.56	20.24	0.16	9.76	0.03
5	95.89	50.00	57.89	6.25	33.33	0.78	21.95	0.20
6	97.26	60.00	69.08	12.96	48.81	2.80	36.59	1.01
7	98.17	70.00	77.63	24.01	59.52	8.24	51.22	4.04
8	99.54	80.00	85.53	40.96	67.86	20.97	60.98	13.42
9	100.00	90.00	92.11	65.61	85.71	47.83	80.49	38.74
10	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00

Table 1: Accuracy results for sequence lengths of 1, 4, 7, and 9 compared to baseline results. All percentages are rounded to two decimal places. Big Brother overall outperforms the baseline, but difficulty grows considerably as sequence length increases.

Sequence Length (n)	Average Max Rank (0, -45, -90)	Average Max Rank (0, -30, -130)	Average Max Rank (0, 55, 100)	Average (all angles)	Std. Dev. (all angles)
1	1.50	1.83	2.06	1.95	0.28
4	3.82	5.98	5.71	5.85	1.18
7	5.43	7.42	7.47	7.45	1.16
9	6.14	7.82	8.13	7.98	1.07

Table 2: Average Max Rank (AMR) for a few representative angles for sequence lengths of 1, 4, 7, and 9. AMR helps estimate Big Brother's performance on a given test dataset; depending on the camera angle, this score can vary depending on camera angle.

fication, which we discuss in the next section.

Despite both training and testing our model on a single test subject, we believe Big Brother's impressive performance on the single digit case does partially validate our hypothesis by showing pen motion, however subtle, can be analyzed to recover *some* information about the text written. Additionally, the success of the single digit case from chained application of relatively unsophisticated techniques for data collection and classification demonstrates this information may be surprisingly accessible. We discuss possible advancements to Big Brother in Section 7.

6.2 Multiple Digit Accuracy

Big Brother's accuracy scores beat the baseline in all multiple digit sequence cases we tested, though performance degrades considerably compared to the single digit case. For $n = 4$, a rank of $r = 9$ gives a likelihood of containing the original digit sequence of 92.11%. This can still offer smaller search space compared to the baseline: 9^4 vs. 10^4, a 34% decrease in size assuming all four digits appear in the nine guesses.

However, failing to achieve 100% accuracy on $r = 9$ demonstrates the model is suceptible to placing the correct digit in the *last ranked position*. This seriously diminishes our overall confidence in our space detection heuristic and iterative ranking procedure described in Section 4.6.2. Ideally, given perfect boundary detection, the model could treat each digit in the sequence as a single independent digit and avoid compounding error. Instead, the classifier is likely discovering several high ranking, locally optimal digit candidates from unintended noise and padding in the path chunks that push the correct digit class out of the way.

Despite Big Brother's middling multiple digit performance, we stress that the reduction in search space could still prove useful. For $n = 9$, taking the top seven ($r = 7$) guesses for each digit gives a search space of 7^9, which will contain the actual nine digit sequence with 51.22% probability, akin to flipping a coin. The baseline offers a 4.04% probability of carrying the nine digit sequence given seven random digits are considered for each digit. A brute force search over nine digits would give a guess magnitude of 10^9, which is $\frac{10^9}{7^9} = 729$ times larger than the selection offered by Big Brother for $r = 7$. This proves Big Brother's suggestions can still prove to be useful when an exhaustive search is needed, placing likely candididates first with appreciable probability.

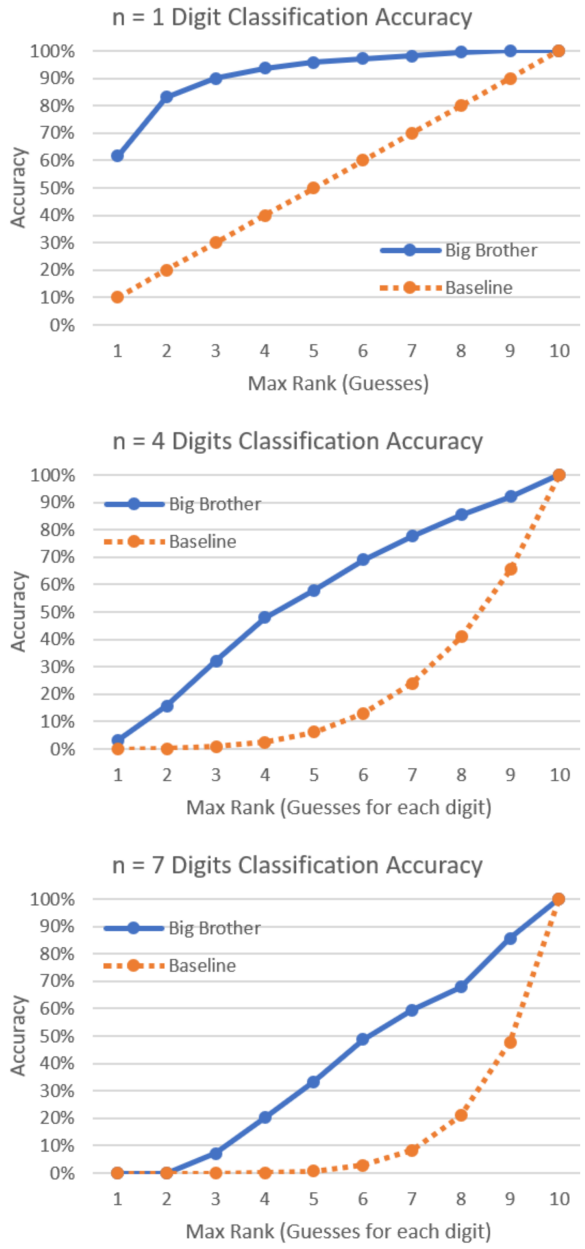

Figure 8: Accuracy scores for sequence length of $n = 1$, $n = 4$, and $n = 7$.

For $n = 1$, out of a set of 219 test samples, Big Brother correctly classifies 61.64% with only a single guess, with the correct class appearing within the top 5 guesses 95.89% of the time. The baseline is given by $\frac{rank}{10}$.

For $n = 4$, out of a set of 152 test samples, Big Brother classifies only 3.29% sequences exactly, but 57.89% of sequences were included in the set of sequences given by five guesses per digit. The baseline is given by $(\frac{rank}{10})^4$.

For $n = 7$, out of a set of 84 test samples, Big Brother considered 33.33% sequences correctly within the top 5 guesses for each digit. The baseline is given by $(\frac{rank}{10})^7$.

7 Limitations and Future Work

In this section, we discuss several of Big Brother's current limitations, and consider improvements that can improve Big Brother's accuracy and usability.

7.1 Data Collection

Due to time and labor limitations, our ground truth data set of 253 single-digit samples consisted of only about 25 examples per digit class, recorded from a single test subject (Patil). A significant drawback to our data collection techniques was the involvement of a digit discretization procedure which reqiured a *manual* frame-by-frame analysis to determine the start and end of each digit. An automated system for highly precise digit detection, with perfect information, would have dramatically improved our data collection rate. For example, using an augmented pen like Dodecapen [16] to provide more information when preparing our model could enable automatic path discretization.

Additional ground truth data, from a much wide selection of subjects, would allow the algorithm to better capture variations in handwriting. Improvements to the *quality* of the ground truth data are also likely to improve performance. Motion tracking is inherently imperfect and noisy; a specialized motion tracking algorithm combined with smoothing techniques can reduce glitches created during data capture.

7.2 Classification

Our current TWED and nearest neighbor classification strategy is efficient, though relatively unsophisticated compared to more complex classification models such as neural networks [12].

Subtle improvements may be gained from simple hyperparameter tuning or changes to our distance formula to better capture features such as writing scale and speed. Our scoring function currently computes the squared distance in both the X and Y component (plus a time penalty). However, in practice, this may cause one component to outweigh the other undesirably. A more sophisticated kernel may prove beneficial, and refinements to DTW/TWED could better distinguish handwritten characters from each other [4].

Our current approach focuses on tracking the rear end of the pen as a single point. However, this ignores the full geometry of the pen. Complete knowledge of the 3D pose of the pen would offer depth information lost from the 2D view of the camera frame. Alternative 3D motion tracking algorithms like those discussed in Section 4.4 could provide dramatically more information for our classifier to digest.

While vision is a salient source of information, there are other side channels present in our security model that may be useful for discriminating digits. For example, the sound of the pen as it contacts the paper could be used to perform segmentation. Our current segmentation strategy depends on our greedy space detection heuristic discussed in Section 4.6.1, which only estimates the beginning of each space (and the end of each digit). Better knowledge of digit boundaries would remove the need to iterate over digit start positions, removing an unknown dimension from our classification and ranking procedure.

7.3 Outputs

Big Brother currently outputs a top ten ranking of digits for each digit in a predicted sequence, and our accuracy scoring metric reflects what "max rank" is required to achieve a probability of *containing* the correct seqeunce within the rankings. We showed in Section 6.2 that this output scheme does help reduce overall search space, but the guess magnitude still remains exponentially large.

As discussed earlier, weaknesses in our classification strategy prevent Big Brother from treating digits independently and offering an overall accuracy score for each complete sequence. Ideally, Big Brother would output a ranking of *sequences* ordered by probability, so the attacker would not have to randomly pick among a random probabilistic space. This would also help the attacker in scenarios where limited attempts are available, e.g. a smartphone which initiates a lockdown after several failed PIN entries.

7.4 Usability

Our current threat model assumes the attacker has access to certain conveniences such as a fixed camera position and paper location, clear visibility of the rear end of the pen, and knowledge of the number of digits written. We presume a more powerful computer vision model could compensate for camera or paper movement by automatically estimating the position of both throughout the video. Our early vision prototypes experimented with the use of color space tracking to detect the pen's position like the model proposed by Seok. et al. [14]. A color and shape matching approach would work especially well for detecting paper, which typically contrasts its surroundings and is rectangular in shapw. Furthermore, improvements to our classifier and discretization methods may naturally erase the need to provide the quantity of digits.

8 Conclusion

In this paper, we present a novel information stealing attack against pen-and-paper. We demonstrate that pen motion is a salient source of data for inferring written text, and present Big Brother, a system that allows an adversary to infer what a victim writes from camera footage of the victim writing, even when the writing surface is not clearly visible.

While our system does not guarantee identification of the sequence of written digits exactly, it outputs a space of possibilities that are likely to include the original sequence. This cuts the state space of possible guesses significantly, reducing the average time required for performing an exhaustive search. Combined with a small amount of other information an attacker may have (such as the last four digits of a phone number or SSN), this capability could allow an attacker to further expand the surface area of their knowledge.

Given we are not domain experts in the field of computer vision, we believe a knowledgeable adversary, such as a large agency or research group, is likely to improve on these results significantly. Such improvements would almost certainly present a real risk to the security of pen and paper in the presence of camera surveillance.

References

[1] Form w-9 (rev. october 2018) - fw9.pdf. https://www.irs.gov/pub/irs-pdf/fw9.pdf. Accessed: 12-05-2018.

[2] Opencv library. https://opencv.org/. Accessed: 12-08-2018.

[3] ASONOV, D., AND AGRAWAL, R. Keyboard acoustic emanations. In *IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004* (May 2004), pp. 3–11.

[4] BASHIR, M., AND KEMPF, J. Reduced dynamic time warping for handwriting recognition based on multidimensional time series of a novel pen device. *International Journal of Intelligent Systems and Technologies, WASET 3*, 4 (2008), 194.

[5] CAI, L., AND CHEN, H. Touchlogger: Inferring keystrokes on touch screen from smartphone motion. *HotSec 11* (2011), 9–9.

[6] KRISTAN, M., MATAS, J., LEONARDIS, A., FELSBERG, M., CEHOVIN, L., FERNANDEZ, G., VOJIR, T., HAGER, G., NEBEHAY, G., AND PFLUGFELDER, R. The visual object tracking vot2015 challenge results. In *Proceedings of the IEEE international conference on computer vision workshops* (2015), pp. 1–23.

[7] LEPETIT, V., FUA, P., ET AL. Monocular model-based 3d tracking of rigid objects: A survey. *Foundations and Trends® in Computer Graphics and Vision 1*, 1 (2005), 1–89.

[8] LI, S., XU, C., AND XIE, M. A robust o(n) solution to the perspective-n-point problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence 34* (2012), 1444–1450.

[9] LUKEZIC, A., VOJÍR, T., CEHOVIN, L., MATAS, J., AND KRISTAN, M. Discriminative correlation filter with channel and spatial reliability. *CoRR abs/1611.08461* (2016).

[10] MARTEAU, P.-F. Time warp edit distance with stiffness adjustment for time series matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence 31*, 2 (2009), 306–318.

[11] MUNICH, M. E., AND PERONA, P. Visual input for pen-based computers. In *Image Processing, 1996. Proceedings., International Conference on* (1996), vol. 2, IEEE, pp. 173–176.

[12] RAJAVELU, A., MUSAVI, M. T., AND SHIRVAIKAR, M. V. A neural network approach to character recognition. *Neural networks 2*, 5 (1989), 387–393.

[13] SALVADOR, S., AND CHAN, P. Toward accurate dynamic time warping in linear time and space. *Intell. Data Anal. 11*, 5 (Oct. 2007), 561–580.

[14] SEOK, J.-H., LEVASSEUR, S., KIM, K.-E., AND KIM, J. Tracing handwriting on paper document under video camera. ICFHR.

[15] TRIER, O. D., JAIN, A. K., TAXT, T., ET AL. Feature extraction methods for character recognition-a survey. *Pattern recognition 29*, 4 (1996), 641–662.

[16] WU, P.-C., WANG, R., KIN, K., TWIGG, C., HAN, S., YANG, M.-H., AND CHIEN, S.-Y. Dodecapen: Accurate 6dof tracking of a passive stylus. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology* (2017), ACM, pp. 365–374.

[17] YASUDA, K., MURAMATSU, D., SHIRATO, S., AND MATSUMOTO, T. Visual-based online signature verification using features extracted from video. *Journal of Network and Computer Applications 33*, 3 (2010), 333–341.

[18] YILMAZ, A., JAVED, O., AND SHAH, M. Object tracking: A survey. *ACM Comput. Surv. 38*, 4 (Dec. 2006).