Sketcha: A Captcha Based on Line Drawings of 3D Models

Steven A. Ross
Princeton University
Princeton, NJ, USA
saross@cs.princeton.edu

J. Alex Halderman
University of Michigan
Ann Arbor, MI, USA
jhalderm@eecs.umich.edu

Adam Finkelstein
Princeton University
Princeton, NJ, USA
af@cs.princeton.edu

ABSTRACT

This paper introduces a captcha based on upright orientation of line drawings rendered from 3D models. The models are selected from a large database, and images are rendered from random viewpoints, affording many different drawings from a single 3D model. The captcha presents the user with a set of images, and the user must choose an upright orientation for each image. This task generally requires understanding of the semantic content of the image, which is believed to be difficult for automatic algorithms. We describe a process called *covert filtering* whereby the image database can be continually refreshed with drawings that are known to have a high success rate for humans, by inserting randomly into the captcha new images to be evaluated. Our analysis shows that covert filtering can ensure that captchas are likely to be solvable by humans while deterring attackers who wish to learn a portion of the database. We performed several user studies that evaluate how effectively people can solve the captcha. Comparing these results to an attack based on machine learning, we find that humans possess a substantial performance advantage over computers.

Categories and Subject Descriptors

K.6.5 [**Management of Computing and Information Systems**]: Security and Protection—*Authentication*; K.4.4 [**Computers & Society**]: Electronic Commerce—*Security*

General Terms

Design, Experimentation, Human Factors, Security

Keywords

security, CAPTCHA, 3D models, drawings

1. INTRODUCTION

This paper introduces a captcha [2] called "Sketcha" based on line drawings created from 3D models. Sketcha requires the user to rotate each image in a set of drawings until every one is upright, by clicking to turn them 90-degrees at a time (Figure 1). The set is selected randomly from a pool of drawings rendered from 3D models in a large database. Using randomized viewing parameters, many different images can be rendered from a single 3D model. People are better than machines at recognizing and understanding images of 3D shapes (at least until the general problem of computer vision is solved). Furthermore, the use of line drawings preferentially obfuscates the objects, like the distortions employed in text-based captions, potentially broadening the relative gap between human recognition and that of automatic algorithms. Moreover, one study suggests that people can recognize drawings *faster* than photographs, and with equal accuracy, at least in the case of pictures of human faces [11].

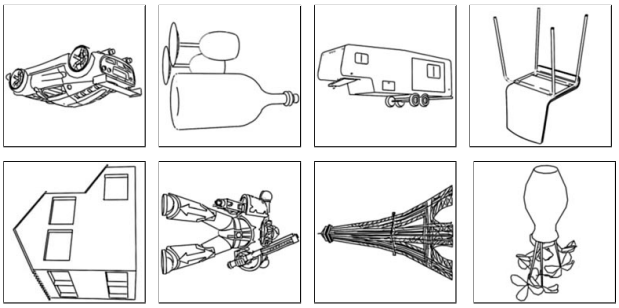


Figure 1: Example captcha based on line drawings. The user's goal is to rotate each image until it is upright, choosing among four orientations by clicking on the image. Each line drawing was automatically rendered from a 3D model using a randomized point of view, providing for many possible images from each model.

Captchas exploit the gap between what humans and machines can accomplish; any simple puzzle that humans can solve well but that is considered to be difficult for computers may form the basis for a captcha. The most prevalent captchas are based on an image containing text that has been obfuscated by a variety of distortions (warping, image noise, overlapping letters, overdrawn lines and other shapes, etc.) The designer must choose a degree of obfuscation which makes it very unlikely that an adversarial program can deduce the text. At the same time the text should not be too obfuscated – it should be very likely that the human should be able to recognize the text. Some people find current text-based captchas annoyingly difficult. Luis von Ahn, one of the inventors of captchas, offers the rule of thumb that humans will tolerate a test that they can solve about 9 out of 10 times; if the test is more difficult for humans, frustration will deter them from using the service behind the captcha [1].

WWW 2010, April 26–30, 2010, Raleigh, North Carolina, USA.
ACM 978-1-60558-799-8/10/04.

Long-standing problems in AI offer good resources for captcha designers: we believe an adversary will not be able to solve the problem with greater accuracy than techniques previously investigated by the research community [2, 3]. But there remains an arms race between, on one side captcha designers, and on the other side both researchers and hackers. Berkeley researchers Mori and Malik [17] were able to defeat the text-based Gimpy captcha in use by Yahoo in 2002. Last year security experts announced they believe that a European hacker has compromised the text-based captcha in use by Google [18]. As attackers' methods surpass the abilities of the least-capable humans, the problem (e.g. reading obfuscated text) can no longer generate a captcha, and designers need to turn to new problems.

Our approach follows that of Gossweiler et al. [12], who introduced the idea of image *orientation* forming the basis for a captcha. Their approach, called the "What's Up" captcha, uses images drawn from popular web searches as a potentially huge database. This design enjoys several nice properties, including simplicity, language independence, and the web as an ever-growing resource for database images.

Gossweiler et al. mention the possible extension of their method to use 3D models (the basis for our captcha) which offer several potential advantages as a source of imagery. Images selected from the web (the most obvious source for a huge database) are subject to reverse-indexing, for example the service offered by TinEye.com.[1] In contrast, by providing *renderings* (especially line drawings) we offer little support for an attacker to recover the original 3D model. To recognize a previously seen model, the attacker must match any possible rendering (from any angle) against it [9]. Moreover, 3D models have the potential to support a variety of rendering *styles* (e.g. Figure 7) to further obfuscate the image, though in this paper we only study simple line drawings. Models can even be generated *programmatically*, constructing variety of shapes within a family, like buildings or plants, based on random parameters, combinations and arrangements. The computer vision literature offers many tools to the potential attacker of a photo-orientation captcha, for example face detection, sky detection, landscape scenes, and so forth [15, 19]. As observed by Gossweiler et al., the captcha designer can incorporate such tools into the formation of the database and thereby ameliorate the threat of such attacks, and the same principle applies in the 3D case as well. Nevertheless, we believe the gap is broader between what people and computers can currently achieve with regard to recognizing the contents of line drawings. Finally, serving images of models, rather than the models themselves, offers an advantage aside from security, wherever intellectual property restrictions prevent redistribution of the database.

The other major difference is that the What's Up captcha uses continuous rotation (requiring the user's answer be within some tolerance of the correct orientation), whereas Sketcha offers the user only four orientation options. Some benefits of the latter interface are that it is relatively simpler to describe and understand, it is easily implemented in major web browsers, and the task can be accomplished by only taps or mouse clicks. Chow et al. [5] have argued for using captchas that can be performed by pointer clicks, citing speed and simplicity on mobile devices.

[1]In informal testing, we found that TinEye locates the correctly-oriented original for at least one of the example images from the Gossweiler et al. study (Figure 6-6 in [12]).

Researchers have investigated several forms of captchas based on understanding natural images. Warner proposed a system called "KittenAuth" based on the ability to recognize kittens in photographs [22]. Such schemes are known to be weak because the full database of images can be learned by an adversary. Thus, the Asirra system of Elson et al. [7] uses a huge database of photos of cats and dogs (from Petfinder.com), under the assumption that the full database is too large to be learned. (In fact, such a method is often called a HIP for "Human Interactive Proof" rather than captcha, because the latter technically requires that all algorithms and data are publicly known.) However, Golle [10] showed that the Asirra captcha is vulnerable to machine learning attacks; simply put, it is possible to design an algorithm that can identify cats and dogs with enough reliability that the captcha can be solved with probability 0.1, which is sufficiently often to render it ineffective as a security mechanism.

Previous systems have considered the use of 3D models in captchas. Kaplan [14] offered an early proposal based on manual labeling of models that is unlikely to scale due to manual effort in modeling and labeling parts. The web site `www.yuniti.com` uses captchas based on Lambertian renderings of 3D models, but it does not appear to have been subjected to a rigorous security analysis and in fact appears to be susceptible to attack using basic computer vision techniques. Fu et al. [8] describe a method for orienting 3D models of man-made objects that might be used for attacking captchas. Fortunately, our captcha displays *images* rendered from models, rather than the models themselves. Mitra et al. [16] propose highly abstract "emergence images" rendered from 3D models as a potential source of captcha, but offer limited security analysis.

Section 2 describes how we built a prototype of the Sketcha captcha, populating a database with hundreds of models and thousands of images. This section also introduces a process we call *covert filtering* whereby the database can be continually refreshed. Section 3 presents the results of several user studies (involving hundreds of subjects on the Amazon Mechanical Turk) to evaluate people's abilities to solve this puzzle, concluding that for a reasonable range of parameters this is a viable approach. Section 4 considers two broad forms of attack on such a system – both learning a portion of the database by repeatedly querying the system and machine learning attacks. We show that our covert filtering process effectively thwarts an attacker who seeks to learn the pool of images. Finally, we evaluate our test database in the context of a machine learning attack and argue that it would be difficult to close the gap between humans' and machines' ability to solve the captcha.

Thus, the contributions of this paper are:

- a captcha based on orienting drawings derived from 3D models,
- a working prototype, available at `www.sketcha.net`
- the results of several large usability studies,
- a way to continually update a database of such images using "covert filtering,"
- analysis showing that covert filtering resists attackers who try to learn the database,
- and analysis of a machine learning attack.

2. THE SKETCHA CAPTCHA

This section describes our proposed captcha and presents the details of a prototype implementation. First we address the user interface, followed with a discussion of how we produce and maintain a database of 3D models and resulting images.

2.1 Interface

Our captcha requires the user to rotate a series of images until each one is upright. In our implementation, each image is shown as a very small (80x80) thumbnail with one larger (240x240) image that magnifies any image that the mouse hovers over, much like in the Asirra captcha of Elson et al. [7].

Since the images shown are drawings of 3D objects, the viewer must generally recognize the objects in order to understand what is their proper orientation (although sometimes one can make a good guess based on an overall impression of the *kind* of object). To rotate an image by 90° the user simply clicks on it, so there are four orientations to choose from. If the series contains n images there are 4^n combinations, for example 65,536 from only eight images. It takes an average of 1.5 clicks to orient an image, so 12 clicks are expected for an 8-image captcha.

This interface allows the images to be served by the web server in a single (random) orientation, and then client-side javascript rotates the image as the user clicks. (Our implementation works in several common browsers including Firefox, Safari and Internet Explorer.) Thus, the bandwidth requirements are low per captcha, as only one orientation need be sent per image. In our implementation, ten 240x240 images with an average file size of about 12kb each are sent from the server per captcha. This is one practical advantage to choosing 4 discrete possible orientations per image, in contrast to the captchas described by Gossweiler et al. [12] which allow for continuous rotation, and thus either requires more sophisticated software running at the client side (e.g. Flash) to handle rotation or sending many pre-rotated images. Finally, we believe that clicking through only 4 choices is easier to understand and manipulate than continuous rotation.

2.2 Database

To generate images for the captcha, we randomly select models from a large database of models, using randomly chosen viewing parameters. The camera angles range from 60° above the horizon to 40° below the horizon, based on our empirical observations that it is often difficult to orient an image when the camera angles are too close to the north and south poles, and that this effect is stronger from below than above. In Section 3 we present data that supports these observations.

After choosing a random camera angle, our process renders an image using the automatic line drawing system of Cole et al. [6] which offers control of line density even where models are very detailed in some areas. Next we crop the images to the bounding box of the lines, and finally scale the image to 240x240.

Key to the success of these captchas is that the set of all possible images, together with their proper answers, should be difficult or impossible to learn. In one form of attack sometimes called the "Mechanical Turk attack," an adversary pays people a small amount of money (or other incentives) to collect the proper answers for every image in the database. To thwart such adversaries we propose to use three strategies: (1) a large database of models, (2) a constant feed of new models into the database, and (3) varying parameters for different images of a given model. These strategies are discussed below, and an analysis of the conditions under which they are robust is offered in Section 4.1.

One challenge posed by both automatic addition of new models to the database and random selection of views is that some tasks presented to the user might become too difficult for many users. It is possible, for example, that the user would be presented with an image that contains just a few unrecognizable lines. Our solution to this problem is to show a new image to a few people and test whether they orient it consistently, before incorporating it into the database. The way we test these "evaluation" images is to mix a few of them in with the already-vetted images presented in a captcha, randomly, such that a person solving the captcha is simultaneously demonstrating his humanity and testing the evaluation images without knowing which is which. We call this process ***covert filtering***. For example, the user may see 10 images total, where 8 of the 10 are used as a captcha and the other 2 images are evaluation images. If a person correctly solves the captcha based on the 8 vetted images, then we take the given answers for the 2 evaluation images as one person's opinion. Once a certain number of people have consistently oriented an evaluation image, it is inserted into the database. On the other hand if anyone chooses a different orientation it is rejected.

This general approach of using part of a captcha process to do useful work was pioneered by the reCaptcha system of von Ahn et al. [20]. However, rather than using the covert filtering process to achieve an external goal (e.g., interpreting digitized text) we use it to improve the strength of our captcha by growing the database. This framework is also related to the "collaborative filtering" approach of Chew and Tygar [4], who use human input to build captchas based on questions for which there is no "correct" answer. Taking inspiration from their work, we note that it is not really important that people fully recognize the object or even answer correctly, as long as they all agree about the proper orientation of the images. In practice, however, we find that with renderings from 3D models, when people agree about an orientation is almost always the correct upright orientation. Gossweiler et al. [12] also briefly discuss this general approach. However, they did not implement it in their user study, nor did they analyze its security implications. In Section 3 we perform multiple user studies to evaluate this framework with regard to human performance, and in Section 4.1 we analyze the security impact of this strategy.

There are many potential sources of 3D models, including commercial data sets containing many thousands of high-quality models, open source model repositories, and even simply crawling the web for models. We anticipate that as 3D scanning technologies improve, acquiring large model sets will become even easier. The experiments described in this paper have been based on models downloaded from the Google 3D Warehouse. This repository allows people from around the world to upload models for other people to view, share, download, tag, and discuss, much like Flickr.com does for photos. It is currently easier to capture photos than

to create or capture 3D models, so for the moment image databases are much larger and growing more quickly than 3D model databases. Nevertheless, the Google Warehouse contains at least hundreds of thousands of models (Google does not currently report the size) and appears to be growing rapidly. This database has a predominance of buildings, in part because of the ability to geo-locate the model in connection with Google Earth. In selecting models, therefore, we only downloaded models that are not geo-located. In addition, we selected only models with high user ratings. We downloaded 4488 models, randomly, with these criteria.

Not all of these models render well in our line drawing system, for various reasons. For example, objects that are extremely wide and flat, or long and thin, tend not to produce good imagery over the range of views described above, while other models produced mostly-white images for many views. Therefore we eliminated models with extreme aspect ratios, where the ratio of the smallest dimension to the largest dimension was less than 0.1, after which 3851 models survived. Next, our drawing software rendered 20 views of each 3347 models, after which we eliminated models where the average value v of all pixels in the image was too close to white ($v = 1.0$), according to the following criteria. We rejected models where either: $v > 0.99$ in 75% of the images, or $v > 0.995$ in 25% of the images. Of the remaining 2574 models, we selected 400 randomly for the experiments described in Section 3. This entire process was programmatic and therefore in principle could be carried out on an larger scale without human intervention.

We observe that after buildings, the next most prevalent class of models in the 3D Warehouse is cars, or perhaps vehicles. Knowing that a substantial portion of the images in the database come from a particular class of object offers a potential advantage to attackers, who might be able to construct a specialized detector. (By analogy: automatic methods for orienting photos often employ face and sky detectors as these are common in natural images [15, 19].) While our system only takes the first step, by eliminating geo-located buildings, we could use filtering, for example, to limit the number of cars based on the "car" keyword.

3. STUDIES ON HUMAN PERFORMANCE

This section presents the results of three experiments we performed with the prototype implementation described in Section 2, in order to evaluate how effectively people can solve our captcha. We used the Amazon Mechanical Turk as the source of participants in our studies. The Mechanical Turk is a internet service that allows "requesters" (such as researchers) to create small, web-based tasks that may be performed by anonymous "workers." Each worker is typically paid between $0.05 and $0.30 to complete a task. The number of workers on the service is such that particularly attractive tasks are usually undertaken within minutes. Workers on the Mechanical Turk generally seem to favor tasks that take somewhere around 10 minutes to complete.

3.1 Experimental Setup

In each of our studies, the task given was to solve a dozen captchas consecutively, and the data we collected included the orientation each user selected for each image. The first ten captchas presented to a subject contained ten images each, and each of these 100 images were selected randomly from pool of images used in the study, such that two criteria were met: first, in any study a subject would see an image rendered from a particular model no more than once; and second, progress through the overall pool of images used throughout the study was approximately uniform. After the initial ten captchas, two more pages offered an identical interface to the initial ten pages, but repeated 20 of the images shown earlier, selected randomly. This provided a measure of consistency – how many of the repeated images were answered the same way the second time indicates the care with which the user performed the task.

In reporting statistics on selected image orientations, we only considered the initial ten captchas – the final two were used only to measure consistency so as not to bias the statistics towards the repeated images.

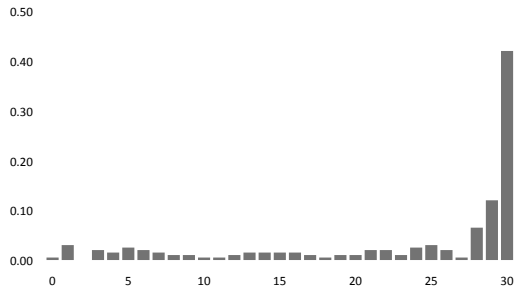
Prior to beginning the task in every study, subjects were given a page containing 10 arrows at random orientations and asked to rotate the arrows so that they all point up, to ensure they understood the basic interface. On each subsequent page the instructions read simply: "Click the images below until they are upright, then click 'Next'." Near the 'Next' button, progress was indicated, for example, by "Page 3 of 12."

After the 12th page, an optional survey asked subjects for gender, age by decade, highest educational degree obtained, and comments. The response rate was high (89%) and these reported:

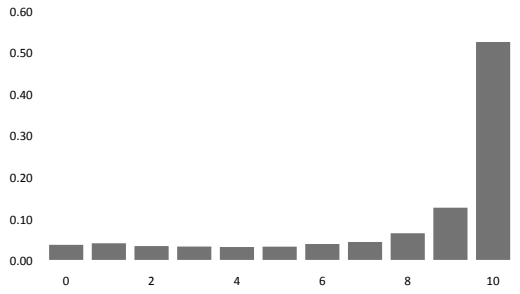
- gender: 54% female, 46% male;
- age: 8% < 20, 42% 20–29, 27% 30–39, 13% 40–49, 8% 50–59, 2% 60–69, 0.3% ≥ 70;
- degree: 4% none, 32% high school, 39% undergrad, 21% graduate, 5% doctorate.

We did not find any significant correlation between overall performance and these attributes, indicating that this task is equally suited to the different groups. While Mechanical Turk workers are likely to be more experienced web users than the general population, there is little reason to expect that they would be better able to interpret and orient line drawings. In addition we used Google Analytics to collect broad demographic information, finding: that our subjects were from 21 countries in total but about 70% were from the US, 20% from India, and the remainder largely from Europe; and that 5 languages were spoken but the vast majority spoke English. We believe these demographics are roughly consistent with the overall pool of workers on the Mechanical Turk. These data are aggregated so we could not compare performance across demographics.

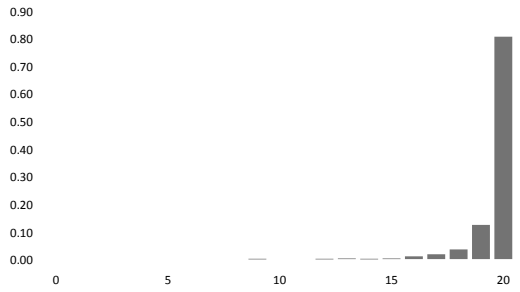
Overall the 558 participants in our studies completed 1,192 tasks (14,304 pages of which 11,920 were test data and the others were duplicate images for verifying consistency in the results). The median time was 8.5 minutes per task. Some workers completed multiple tasks per study (with the constraint that they would see a model no more than once) and some workers completed tasks in multiple studies. We omitted data from 14 completed tasks where the consistency rate was below 12/20 and also 7 where the accuracy on the pool of 100 images was about that of random guessing (indicative of either a misunderstanding or foul play). These data were replaced by that of later subjects.



(a) Study A: 200 models × 1 image × 30 people



(b) Study B: 400 models × 20 images × 10 people



(c) Study C: 200 models × 4 images × 20 people

Figure 2: Distribution of image difficulties in user studies. Horizontal axis encodes number of people (out of a:30 b:10 or c:20) who correctly oriented a particular image – a measure of the difficulty of that image. Vertical axis shows fraction of the images in the study with a given difficulty. Distributions in (a) and (b) are similar, while distribution in (c) includes only images drawn from the rightmost bar in (b).

3.2 Pilot Study

Our first study (Study A) was a pilot experiment designed to collect data about the distribution of difficulties of images in our database. We constructed the database as described in Section 2.2, and then we selected one image from each of 200 models for this study. We tested them with 54 workers performing a total of 69 tasks (690 test captchas), such that every image was seen by at least 30 people.

The fraction of the 30 people who were able to correctly orient a specific image provides a measure of difficulty or ease with with which the image can be oriented. Let us call this measure x_i and take it as an approximate measure of the probability with which an arbitrary new person would be able to orient the image.

Figure 2a shows a histogram of the distributions of x_i over the 200 images in our pilot study. In this plot $f(x_i)$ is the probability density of x_i – the frequency with which we observe x_i, measured with a granularity of 31 bins whose values sum to 1. Notice that the most frequent case is that all 30 people oriented the image correctly, and that this case accounts for 42% of the overall data. This is good news for our proposed captcha, because it means that there are many images for which people can consistently orient them correctly.

The expected value $\mathrm{E}[x]$ over this distribution is

$$\mathrm{E}[x] = \sum_i x_i f(x_i) / \sum_i f(x_i)$$

Unfortunately, the data Figure 2a have $\mathrm{E}[x] = 0.78$ so the likelihood of solving a captcha containing 8 images, for example, is $0.78^8 = 0.14$, in essence an unusable captcha.

These results also suggest that covert filtering might be an effective approach for selecting images that will make a more usable captcha. Suppose we show each image in the pool to 10 people, and eject every image that is incorrectly oriented by at least one person. Of course some moderately difficult images may survive this filter process, but most will not. We can estimate the effect of this filter on the distribution, to the extent that x_i models the likelihood that a new person will be able to correctly orient image i. In particular, the chance that image i will survive the filter is simply x_i^{10}, so the resulting distribution would be $f'(x_i) = x_i^{10} f(x_i)$.

Calculating the expected value over this distribution

$$\mathrm{E}[x'] = \sum_i x_i f'(x_i) / \sum_i f'(x_i)$$

we find that a random image selected from this distribution has a probability $\mathrm{E}[x'] = 0.986$ of being oriented by a new person. Thus a captcha containing 8 such images is expected to be solved with probability 0.89 – a reasonable target rate. This filter forms the basis of our later experiments.

3.3 Filtering Studies

To form a more usable captcha, we will resort to filtering out difficult images as described in Section 3.2. We performed two studies related to this process. In the first (Study B), we began with a larger pool of models and collected distributional statistics as in our pilot study. We used these statistics to select images that were correctly oriented by at least ten participants. We then conducted another study (Study C) using only these filtered images. The results show that covert filtering can significantly improve the ability of humans to solve the captcha.

Study B was based on a (40×) larger pool of 400 models with 20 images each. In this study 504 workers performed a total of 937 tasks (9,370 test captchas). Each image was shown to at least 10 people, the number 10 having been estimated to be sufficient by our analysis of the data from Study A.

The resulting distribution can be seen in Figure 2b. Observe that its shape is similar to that of Figure 2a, albeit with fewer probability values because each image was shown to 10 people rather than 30 so the bins are broader and taller on average. The rightmost data point corresponds to the 52% of the 8000 images for which *all ten* people who saw the image oriented it correctly. The images from this bin form the pool for our next study.

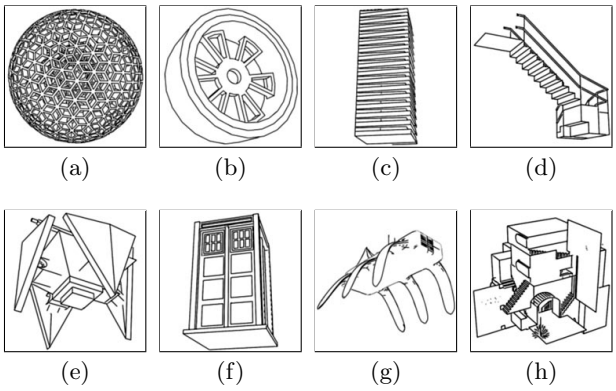


Figure 3: Images removed from the database due to user filtering in Study B. (a-b) recognized but symmetric, (c-d) difficult to recognize, (e-f) typically oriented upside down, (g-h) unfamiliar objects gave rise to incorrect orientations in either (g) bimodal or (h) unimodal distributions. Figure 1 shows example images that survived this filtering process.

In Study C we randomly selected 4 images from each of 200 models, where each image had been correctly oriented by all 10 participants in Study B. Figure 3 show a selection of these filtered images. The test images were shown as a series of captchas, just as in the previous experiments, in this case so that each image was seen by at least 20 people. In this study 98 workers performed a total of 186 tasks (1,860 test captchas).

The resulting probability distribution is shown in Figure 2c. The expected value is $\mathrm{E}[x] = 0.983$, which matches well the value of $\mathrm{E}[x'] = 0.986$ predicted from the data in our prior study as described in Section 3.2. Moreover, $\mathrm{E}[x]^8 = 0.87$ which suggests that this distribution for successful rates of image orientation could be used as the basis for a reasonable captcha.

In addition, we can return to the data for the specific pages of images shown to users and ask: suppose 8 of the 10 images had in fact been a captcha – would the person have succeeded? We find that averaged over all people, all pages, and all subsets of 8 images on each page, that the success rate would be 0.88. This number is slightly better than the rate based solely on the image distribution, because in some cases the user simply pressed the "Next" button without orienting any of the images, which depresses the success rate for images at a higher-than-average rate while only incurring the penalty of a single failed page.

3.4 Discussion

We draw two significant conclusions from these studies. First, the fully-automatic process that randomly selects views and models drawn from the Google 3D Warehouse and renders line drawings from them generates imagery that people can often orient correctly, but not often enough to be used in the kind of captcha proposed herein. Second, the filtering process that rejects images that were incorrectly oriented by at least 1 out of 10 people removes enough of the difficult imagery that the resulting pool can be used for a captcha.

Recall from Section 1 the rule of thumb that people will tolerate a captcha that they can solve 9 out of 10 times they try. Our observed success rate of 0.88 is in the ballpark. Moreover, we have several reasons to believe that in practice the sketcha captcha could have a significantly higher success rate. First, the incentives in our studies do not quite match the incentives of a true captcha. Our workers tended to proceed through a series of captcha pages with two competing goals: to get *most* of the images right (which they are paid to do) and to finish quickly (so they can move onto their next job and make more money). With the proposed captcha, the person's goal is to orient *all* of the images correctly; if they fail they have to try again until they succeed. Therefore, we believe that people would be a little more careful in the real setting. (One could imagine trying to design an incentive structure for the studies on the Mechanical Turk that more closely matched that of a captcha, for example declining to pay people who failed, but we felt this would be unfair.)

In a production captcha system, we would also identify and eject images that survived the initial covert filtering process, but turned out later to have a higher-than-average failure rate, thereby further improving average performance over time. Furthermore, we believe the overall quality of the initial database, prior to covert filtering, could be improved in several ways. For example, by using more sophisticated heuristics that look for difficult models such as those that have strong symmetries such as the wheel shown in Figure 3. Finally, by more narrowly restricting the camera views used in production, we see an opportunity to further improve the initial database. Figure 4 shows the accuracy for the images shown in Study B as a function of the camera angle over the horizon. We see that by restricting the range of angles to the range $[-10°, 50°]$, we could substantially improve the quality of the images in the initial pool.

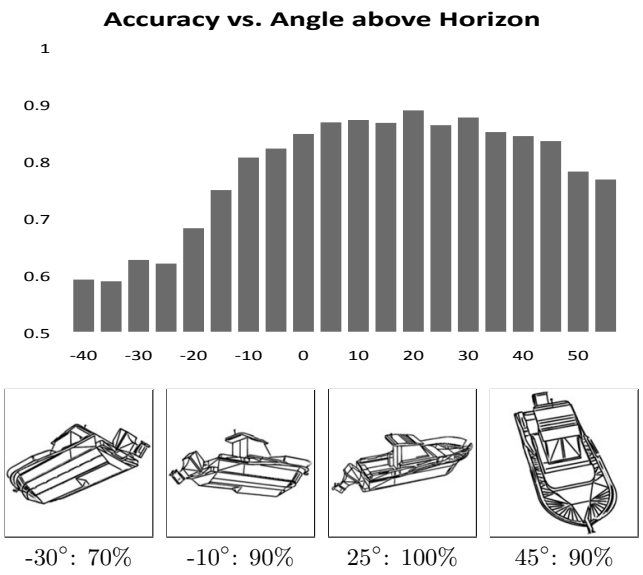


Figure 4: Accuracy by angle above the horizon. People tend to be most accurate when the camera angle is about 20° above the horizon. Views below the horizon (negative angles) have lower accuracy than those above (positive). Label d matches the histogram bin containing values in $[d,d+5]$ degrees. Images labeled with approximate angle:accuracy.

Finally, we note that the median times to complete the 12 pages in Study C (6.5 mins) was significantly lower than that of Study B (8.8 mins). This is not surprising, since many of the difficult images had been removed. These numbers indicate that a person could typically solve a 10-image sketcha captcha in about 35 seconds. Moreover, it might actually be faster as the recorded numbers probably include times in which some workers took breaks. Nevertheless, one limitation of this technique is that this time is probably longer than the time to solve a typical text-based captcha.

4. SECURITY ANALYSIS

In this section we consider possible classes of attack and how they compromise this form of captcha. First we discuss the attacker who concentrates on learning a fraction of the database of images simply by guessing randomly, without regard for the actual image content. The attacker may either assail the system with many guesses in rapid succession to learn some of the database, or may begin by stealing a fraction of the database. Under this form of attack, the system is compromised as the attacker learns enough of the images in the database so as to significantly increase the probability of solving future captchas. Next we investigate an alternate approach wherein the attacker does not bother to remember previously seen images, but rather concentrates on using the content of known images to train a machine learning algorithm for selecting the correct answer for new, previously unseen images. Under this second attack, the system is compromised as the attacker's algorithm increases the chance of correctly solving the captcha significantly above that of random guessing. Finally, we consider an attacker that uses both of these attacks in tandem.

4.1 Database attacks

Here we discuss the conditions under which an attacker compromises the captcha by learning part of the image database via guessing. As the attacker learns more of the database, his chance of guessing the answer to a captcha improve, because he is likely to recognize some of the images. However, we will show that in order to maintain knowledge of any fraction of the database over time, the attacker must sustain a substantial portion of the overall traffic to the database. The dilemma for the attacker is that as he learns more of the database, making it easier for him to guess the captcha, it becomes harder for him to learn new images in order to maintain his rate of knowledge.

Suppose the attacker's rate of traffic represents a fraction α of the overall traffic C to the captcha, and that the remaining fraction $(1-\alpha)$ comes from legitimate users. (Any other non-legitimate traffic, say from other attackers, may be assigned to α for the purposes of this discussion.) The legitimate traffic causes new images to be added to the database at some rate ℓ_h due to covert filtering. If the attacker knows a fraction d of the database, he must learn new images at a rate $d\ell_h$ in order to keep up.

The covert filtering process described in Section 2.2 adds images to the database at the rate:

$$\ell_h = (1-\alpha)Cmq/t \tag{1}$$

where m is the number of images being evaluated in each captcha through covert filtering (2 in our examples), q is the fraction of our pool of evaluation images that survives the covert filtering process (0.52 in Study B) and t is the expected number of times an image must be shown in the covert filtering process before it is either rejected (because someone failed to orient it properly) or it is added to the database. For example, in Study B, images were shown 10 times, but the "mean time to failure" for those images lowered t to 7.3.

Next we consider how quickly the attacker can learn new images by guessing. We observe that if the attacker guesses the answer to a captcha and it is rejected, he learns relatively little – only the fact that at least one of the test images was not correct. On the other hand, if his answer is accepted, then he knows that every test image was correct. Suppose that out of n images he already knew the answer for k of them, and he correctly guessed the other $n-k$. In that case he learned $n-k$ new images. (We can ignore the fact that the attacker does not know which n of the $m+n$ images in the captcha are already in the database and which m are being evaluated; he can simply treat them all as "correct.")

Since the attacker knows fraction d of the database, the probability of his knowing exactly k of the n images in the captcha is given by:

$$p_{nk} = \binom{n}{k} d^k (1-d)^{n-k}$$

Moreover, the probability of his guessing all of the $n-k$ unknown images is g^{n-k} where g is the chance of guessing one ($\frac{1}{4}$ in our interface), and in that case he learns $n-k$ images. Thus, for each attempted captcha he can expect on average to learn:

$$\ell_1 = \sum_{k=0}^{n} g^{n-k}(n-k)p_{nk}$$

Recall that the attacker is attempting captchas at rate αC, and that this must allow him to learn at least as fast as $d\ell_h$, so:

$$\ell_a = \alpha C \ell_1 \geq d\ell_h \tag{2}$$

Equation (1) and inequality (2) place a lower bound on the fraction α of the traffic to the captcha necessary for the attacker to sustain in order to continue knowing a fraction d of the database. Collecting terms it is easy to show that:

$$\alpha \geq 1/\left(1 + \frac{t}{mq}\sum_{k=0}^{n} g^{n-k}(n-k)\binom{n}{k} d^{k-1}(1-d)^{n-k}\right) \tag{3}$$

While inequality (3) is messy, it is easy to evaluate in specific cases, as with the parameters for Sketcha summarized after Equation (1). Figure 5 shows a plot of α as a function of d. If the attacker starts from no knowledge of the database and is trying to learn a fraction of it, he has to climb over the hump from the left side by sustaining a tremendous surge of traffic (95% at the peak). If that is not possible, the attacker remains to the left of the peak and his knowledge of the database offers him only marginal advantage over random guessing.

On the other hand suppose the attacker was somehow able to *steal* the entire database. In this case he has to climb the curve on the right side in order to maintain his knowledge of the database (because as new images are added to the database, he sees them only rarely and thus it is difficult for him to learn at the same rate). So an attacker who obtains the entire database must fall down the curve from the right

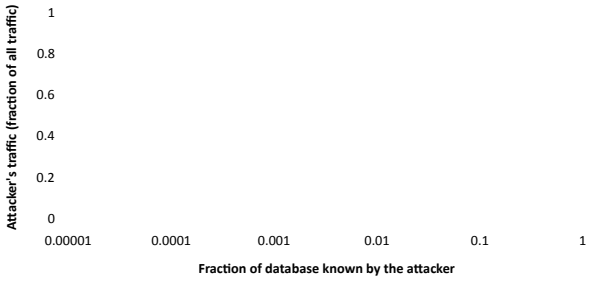


Figure 5: Database Attack. This plot places a lower bound on the fraction of the traffic to the captcha that must come from the attacker (α, vertical axis) as a function of how much the database is already known to the attacker (d, horizontal axis), in the steady state. If the attacker's traffic drops below this bound, the database will grow faster than his learning rate, due to covert filtering. Starting from no knowledge of the database (left side) the attacker must exert 95% of the traffic to the database to climb over the hump. Even if the attacker has managed to learn as much as 80% of the database (right valley), he must sustain 47% of the overall traffic in order to maintain this knowledge.

until his level of traffic can sustain the steady state. If that level is below the minimum of the curve (47% in Figure 5) then he will not be able to learn quickly enough to maintain any fraction of the database and over time his knowledge will dwindle. In this sense the process emerging from covert filtering can be thought of as giving the database a "self-healing" property.

We can also analyze the What's Up captcha of Gossweiler et al. [12] using the same machinery. Their paper suggests that using three images ($n = 3$) provides a reasonable tradeoff between security from attacks and difficulty for humans. Suppose we add a fourth image for evaluation ($m = 1$), and that it takes on average the same number of trials to determine whether or not to add it to the database as in our examples ($t = 7.3$). Their paper suggests that roughly half of the images survive this evaluation process ($q = 0.5$). With these parameters we produce a plot similar in shape to that Figure 5, but with a lower peak (65%) and shallower valley. Thus we conclude that the covert filtering process should also resist database attacks for the What's Up captcha. However, it appears that the scenario in Sketcha where there are more components, each solved more easily, offers better resistance to this form of attack.

Finally, we note that for small values of C, the bound in equation (3) may not be prohibitive for an attacker, which has security implications for web sites wishing to use captchas with covert filtering. If a web site that generates a small amount of traffic maintains its own database of images, an attacker may be able to sustain a high rate of traffic relative to legitimate users. Therefore, covert filtering is effective in contexts where the captcha is implemented centrally and serving many users – in this way sites with low traffic can find "safety in numbers" by sharing a common database.

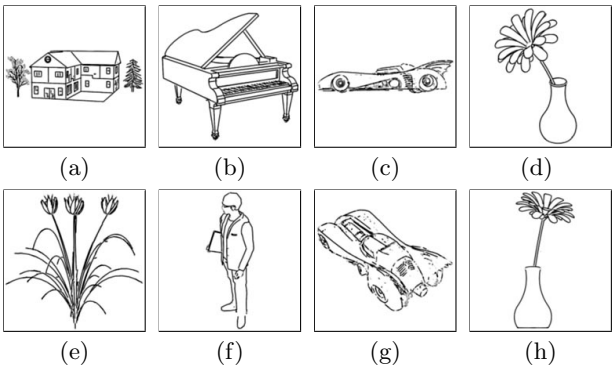


Figure 6: Examples from machine learning attack. We trained a SVM on half of the images in Study B and then evaluated it on the remainder. Objects in the upper row were oriented correctly while the lower row failed. The SVM generally classified boxy objects better than organic forms, and tended to do better for near-horizon views like (c) than off-angle views like (g). Many successes and failures are difficult to explain, such as (d) and (h).

4.2 Machine learning attacks

In addition to explicitly learning the contents of the image database, an attacker may use a machine learning algorithm to build a general-purpose classifier for images of the type used in the captcha. Automatic image orientation detection is a well-studied topic when the subject is a color photograph (e.g., [15, 19]). Current generation algorithms report high (>90%) accuracy in selecting the correct orientation for a general photograph among four 90° rotations. However, we believe that our images are robust against current machine learning methods since line drawings contain less information that photographs. In particular, line drawings lack color, texture, background objects and scenery, as well as high-level semantic cues like grass, sky, buildings, faces, and the like.

Luo and Boutell [15] describe an algorithm for orienting photographs that performs well and is fairly representative of current methods. Their algorithm uses a host of classifiers that leverage the aforementioned image properties like color distribution and semantic features. However, only one of their classifiers makes sense to apply to line drawings: the support vector machine (SVM) based on edge detection histograms, for which they used the technique of Wang and Zhang [21]. Therefore, we implemented the same SVM, which creates feature vectors based on spatial edge detection histograms. Such a histogram is calculated for each block that results from dividing the image into a 5x5 grid, then classifying all pixels according to their edge angle as calculated by the canny edge detection algorithm. For line drawings, we note that a canny edge detector will turn each line into two lines, one in a direction that is rotated 180° from the other. In this way all angles can be expressed in the range 0-180 instead of 0-360, knowing that edge pixels come in pairs. Therefore, our histograms have 19 bins, of which the first 18 are used for edge pixels and the last one is used for pixels that do not correspond to an edge.

To test the strength of our image database against an attack that uses the algorithm described above, we created

sets of feature vectors for each the 800 images used in Study C. Each image produced four sets of feature vectors labeled for each of the four possible orientations. In this way, the SVM correctly classifies an image if it labels according to its proper orientation. We split the data into halves, and trained a multi-class SVM on each independently. One half trained the SVM and we tested it on the other half; then we swapped the halves and repeated the test. The SVM classified images with 61% accuracy on average. Figure 6 shows examples for which it performed well or failed.

These results show that a machine learning algorithm can do significantly better than random guessing. However, an accuracy level of 61% still gives an attacker little hope of breaking the captcha. An attacker using an algorithm with such accuracy would correctly classify eight images in only 1.9% of cases.

There are several defenses against a machine learning algorithm that has high accuracy. One could resort to filter out images that can be solved by particular machine learning algorithms, as proposed by Gossweiler et al.. Pre-filtering images this way would have little impact on the accuracy of humans in completing the task, for several reasons. First, one would need to remove relatively few images to skew the statistics of the classifier towards randomness, so doing this could have little effect on high success rate of humans found in Section 3.3. Second, in looking at the performance of the SVM on this data set we see little correlation with the human performance. The use of rendered line drawings also affords us the ability to vary the rendering process to create images that are targeted at defeating machine learning attacks. The rendering process leaves room for extensive stylization and obfuscation of the object that could confuse a machine learning algorithm based on edge distribution, but can be made in such a way as to preserve the semantic meaning of the image for a human observer.

Finally, we consider the case where an attacker *combines* the database and machine learning attacks discussed in Sections 4.1 and 4.2. The analysis leading to equation (3) supposes that an attacker who does not know an image in the database guesses it with probability equal to random guessing ($\frac{1}{4}$). However, if the attacker uses machine learning to gain advantage in this guess, one might worry that he would be able to learn the database with a much lower traffic rate than emerged from the analysis in Section 4.1. In this situation, the hump on the left of the plot in Figure 5 is attenuated, and therefore the attacker can climb it on the left, but will need to sustain a level equal to about 10% of the legitimate traffic in order to keep up with the growing database. Obviously this attacker does better than one without the aid of machine learning, but in many contexts this remains a prohibitive barrier for all but the most resourceful attackers.

5. CONCLUSION AND FUTURE WORK

This paper presents the Sketcha captcha, a task which requires users to determine the upright orientation for a selection of 3D objects rendered as line drawings. By leveraging a large database of common objects, we render a collection of images from various angles to use in the task. We apply covert filtering to ensure that the images used in the captcha can be solved by humans with high accuracy. In addition, a production implementation would actively add new images to the database to thwart attackers.

Figure 7: Stylization. In addition to varying viewpoints, a single 3D model can be rendered with a broad range of stylization using methods such as that of Kalnins et al. [13].

We believe that line drawings of 3D models are a source of images that is stronger against machine learning attacks than previously suggested image-based captchas. Compared to photographs, line drawings lack detail and cues such as color, leaving less information for computers, but our studies show that this does not make them prohibitively difficult for users to orient.

We ran three user studies to test our captcha and filtering approaches. The results show that we can use covert filtering to increase the human rate of success sufficiently for the task to serve as a practical captcha.

We tested the viability of machine learning attacks by implementing a support vector machine. It was able to orient our test images with modest accuracy, but its performance was insufficient to break the captcha. Machine learning techniques may improve in the future, but our system can adapt by pre-filtering the database to remove images that are successfully oriented by such methods or by changing the image rendering process until the performance of the orientation algorithms drop.

This project suggests a number of areas for future work, including:

- **Obfuscations available for 3D.** In this paper we rendered models only with a simple line drawing style. However, there are many styles available, even within the realm of line drawings (Figure 7). We would like to explore a range of techniques available for further obfuscating the images, hopefully thwarting machine learning algorithms without adversely affecting human performance. For example, we could use wiggly lines rather than straight ones, or we could randomly add lines to the image that are uncorrelated with the rest of the drawing.

- **Other tasks.** In this paper we used rotation as the goal, but there are many other tasks that could be given, based on semantic understanding of the drawing. For example, we could ask people to match images drawn from the same model but with different rendering parameters.

- **Deployment.** Our user studies have been quite extensive, but performed in an artificial setting where users were paid to solve a task. We would like to study this captcha in the context of a working web site where visitors have the *actual* captcha experience.

Acknowledgments

We would like to thank Brian Brewington, Rich Feit, Mark Limber, and the Google 3D Warehouse for the models used in this paper as well as helpful guidance in the project. We are grateful for the encouragement and advice of Luis von Ahn. We also thank Forrester Cole for support in adapting his "dpix" automatic line drawing software, and Mark Gray for an early prototype based on photos. This work was sponsored in part by a Google Research Award.

6. REFERENCES

[1] Luis von Ahn. Personal communication, 2008.

[2] Luis von Ahn, Manuel Blum, and John Langford. CAPTCHA: Using hard AI problems for security. In *Proceedings of Eurocrypt*, pages 294–311. Springer-Verlag, 2003.

[3] Luis von Ahn, Manuel Blum, and John Langford. Telling humans and computers apart automatically. *Communications of the ACM*, 47(2):56–60, 2004.

[4] Monica Chew and J. D. Tygar. Collaborative filtering captchas. In Henry S. Baird and Daniel P. Lopresti, editors, *HIP*, volume 3517 of *Lecture Notes in Computer Science*, pages 66–81. Springer, 2005.

[5] Richard Chow, Philippe Golle, Markus Jakobsson, Lusha Wang, and XiaoFeng Wang. Making captchas clickable. In *HotMobile '08: Proceedings of the 9th workshop on Mobile computing systems and applications*, pages 91–94, 2008.

[6] Forrester Cole, Doug DeCarlo, Adam Finkelstein, Kenrick Kin, Keith Morley, and Anthony Santella. Directing gaze in 3D models with stylized focus. *Eurographics Symposium on Rendering*, pages 377–387, June 2006.

[7] J. Elson, J. Douceur, J. Howell, and J. Saul. Asirra: a CAPTCHA that exploits interest-aligned manual image categorization. In *Proceedings of ACM CCS 2007*, pages 366–374, 2007.

[8] Hongbo Fu, Daniel Cohen-Or, Gideon Dror, and Alla Sheffer. Upright orientation of man-made objects. *ACM Trans. Graph.*, 27(3), 2008.

[9] Thomas Funkhouser, Patrick Min, Michael Kazhdan, Joyce Chen, J. Alex Halderman, David Dobkin, and David Jacobs. A search engine for 3D models. *ACM Trans. Graph.*, 22(1):83–105, 2003.

[10] Philippe Golle. Machine learning attacks against the Asirra captcha. Technical Report 2008/126, IACR Cryptology ePrint Archive, 2008.

[11] Bruce Gooch, Erik Reinhard, and Amy Gooch. Human facial illustrations: Creation and psychophysical evaluation. *ACM Trans. Graph.*, 23(1):27–44, 2004.

[12] Rich Gossweiler, Maryam Kamvar, and Shumeet Baluja. What's up captcha? A captcha based on image orientation. In *Proceedings of WWW 2009, the 18th International World Wide Web Conference*, 2009.

[13] Robert D. Kalnins, Lee Markosian, Barbara J. Meier, Michael A. Kowalski, Joseph C. Lee, Philip L. Davidson, Matthew Webb, John F. Hughes, and Adam Finkelstein. WYSIWYG NPR: drawing strokes directly on 3D models. *ACM Transactions on Graphics*, 21(3):755–762, July 2002.

[14] Michael Kaplan. The 3-D CAPTCHA. http://spamfizzle.com/CAPTCHA.aspx.

[15] Jiebo Luo and Matthew Boutell. Automatic image orientation detection via confidence-based integration of low-level and semantic cues. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(5):715–726, 2005.

[16] Niloy J. Mitra, Hung-Kuo Chu, Tong-Yee Lee, Lior Wolf, Hezy Yeshurun, and Daniel Cohen-Or. Emerging images. *ACM Transactions on Graphics*, 28(5), 2009.

[17] Greg Mori and Jitendra Malik. Recognizing objects in adversarial clutter: Breaking a visual captcha. In *Computer Vision and Pattern Recognition CVPR03*, pages 134–141, 2003.

[18] Brad Stone. Breaking Google captchas for some extra cash. *New York Times*, March 13, 2008.

[19] Aditya Vailaya, Hongjiang Zhang, Senior Member, Changjiang Yang, Feng-I Liu, and Anil K. Jain. Automatic image orientation detection. *IEEE Transactions on Image Processing*, 11:600–604, 2002.

[20] Luis von Ahn, Benjamin Maurer, Colin McMillen, David Abraham, and Manuel Blum. reCAPTCHA: Human-Based Character Recognition via Web Security Measures. *Science*, 321(5895):1465–1468, 2008.

[21] Yongmei Wang and Hongjiang Zhang. Content-based image orientation detection with support vector machines. *IEEE Workshop on Content-Based Access of Image and Video Libraries (CBAIVL 2001)*, pages 17–23, 2001.

[22] Oli Warner. KittenAuth. http://www.thepcspy.com/kittenauth.