Breaking the Clock Face HIP

Zhenqiu Zhang [†], Yong Rui[‡], Thomas Huang [†] and Cem Paya[‡]
[†]Beckman Inst. for Advance Technology
University of Illinois at Urbana-Champaign
{zzhang6, huang}@ifp.uiuc.edu

[‡]Microsoft Corporation
Redmond, WA 98052
{yongrui, cemp}@microsoft.com

ABSTRACT

Web services designed for human users are being abused by computer programs (bots). The bots steal thousands of free email accounts in a minute, participate in online polls to skew results, and irritate people in chat rooms. These real-world issues have recently generated a new research area called Human Interactive Proofs (HIP), whose goal is to defend web services from malicious attacks by differentiating bots from human users. For a HIP challenge to be effective, it needs to be both easy for human and robust against bots attacks. Recently, there is a new HIP design which is based on telling time from a clock face. This is a very innovative idea and quite easy for human to pass. However, its robustness to attacks needs verification. In this paper, we present an algorithm that can break the clock face HIP at 87.4%.

1. INTRODUCTION

Web services are increasingly becoming part of people's everyday life. For example, we use free email accounts to send and receive emails; we use online polls to gather people's opinion; and we use chat rooms to socialize with others. But all these Web services designed for human use are being abused by computer programs (bots). Malicious programmers have designed bots to register thousands of free email accounts every minute. Bots have been used to cast votes in online polls. Chat rooms and online shopping are being abused by bots as well [1][3].

The above real-world issues have recently generated a new research area called Human Interactive Proofs (HIP), whose goal is to defend services from malicious attacks by differentiating bots from human users. The first idea related to HIP can be traced back to Naor who wrote an unpublished note in 1996 [8], and the first HIP system in action was developed in 1997 by researchers at Alta Vista [2]. After that, CMU and PARC actively developed several HIPs [1][3][4].

The CMU team so far has been one of the most active teams in HIP, and we highly recommend readers to visit their web site at http://www.captcha.net. One of the HIPs they developed is called Gimpy. Gimpy picks seven random words out of a dictionary, distorts them and renders them to users. The user needs to recognize three out of the seven words to prove that he or she is a human user. The CMU team also developed an easier version, EZ Gimpy, which is currently used at Yahoo's website.

Before any HIP can be put into practice, it has to be very easy for human to use and extremely robust to attacks. A set of detailed HIP design criteria is discussed in [10]. So far, most of the existing HIPs have been based on distorted text, e.g., [1][2][3][4]. The assumption is that human can read distorted text but it is hard for a machine (bots) to read. The text-based HIPs were successful until Berkeley researchers proposed a new object recognition algorithm in July 2003 which breaks EZ Gimpy at 92% and Gimpy at 33% [7]. Researchers then started to explore non-text HIPs. One of the most noticeable new HIPs is the clock face HIP (CFHIP) [6]. It satisfies many of the criteria of being a promising HIP [10]. For example, while texts are different across alphabets (Hebrew, Arabic, English, etc.), clock is culturally universal, which saves significant localization effort in practice. CFHIP works as follows. Per each user request, it automatically synthesizes an image with a clock face embedded in a cluttered background. The user is asked to first find the clock face and then enter the time. If the user can correctly enter the time, CFHIP concludes the user is a human; otherwise, the user is a machine.

HIP is a unique research area in that it creates a win-win situation. If attackers cannot defeat a HIP algorithm, that algorithm can be used to defend Web services. On the other hand, if attackers defeat a HIP algorithm, that means they have solved a hard AI problem, thus advancing the AI research. As vision researchers, we take up the challenge of defeating the CFHIP. The rest of the paper is organized as follows. In Section 2, we describe the detailed algorithm on how to break the CFHIP at 87.4%. In Section 3, we present experimental results using 1,000 CFHIP challenges. We give concluding remarks in Section 4.

2. ALGORITHM TO DEFEAT CFHIP

To confuse bots, CFHIP images are constructed with complex background. For example images, see Figure 2 and see http://www.ifp.uiuc.edu/~zzhang6/image.zip. The location, shape, color, and intensity of the clock faces are all randomized in different images. In order to read time from a clock face, we need to 1). first detect the location of

the clock face; 2). then detect clock center, tick marks, and clock hands; and 3) finally read the time. In this section, we present detailed algorithm to achieve this goal.

2.1 AdaBoost learning for clock detection

Clock detection in clutter background can be defined as a clock/non-clock classification problem. A clock detector is learned from clock/non-clock training examples using AdaBoost algorithm. AdaBoost has been very successful in object/pattern recognition. For example, it achieves one of the best face detection results [12]. The clock detection task is the following: given an input image, sub-windows at all locations and scales are scanned, and then classified into clock and non-clock objects.

For two class problem, AdaBoost can be described as follows. Let a set of N labeled training examples be $(x_1, y_1), \ldots, (x_N, y_N)$, where $y_i \in \{+1, -1\}$ is the class label associated with example x_i. For clock detection, x_i is an image sub-window of a fixed size (*e.g*, 60x60 pixels) containing an instance of the clock $(y_i = +1)$ or non-clock $(y_i = -1)$ pattern. In the notion of RealBoost (a real-valued version of AdaBoost as opposed to the original discrete one, see Figure 1), a stronger classifier is a linear combination of M weak classifiers.

$$H_M(x) = \sum_{m=1}^{M} h_m(x) \tag{1}$$

where $h_m(x) \in \Re$ are the weak classifiers. The class label for a test x is obtained as $H(x)$ = sign $[H_M(x)]$ (an error occurs when $H(x) \neq y$) while the magnitude $|HM(x)|$ indicates the confidence.

In boosting learning [5][11], each example x_i is associated with a weight w_i, and the weights are updated dynamically using a multiplicative rule according to the errors in previous learning so that more emphasis is placed on those examples which are erroneously classified by the weak classifiers learned previously. This way, the new weak classifiers will pay more attention to those examples. The stronger classifier is obtained as a proper linear combination of the weak classifiers.

0.(Input)
- *(1) Training examples* $z = \{(x_1, y_1), \ldots, (x_N, y_N)\}$, *where* $N = a + b$; *of which* a *examples have* $y_i = +1$ *and* b *examples have* $y_i = -1$;
- *(2) The number* M *of weak classifiers to be combined;*

1.(Initialization)
- $w_i = 1/2a$ *for those examples with* $y_i = +1$
- $w_i = 1/2b$ *for those examples with* $y_i = -1$

2.(Forward Inclusion)

For $m = 1, \ldots, M$:
- *(1) Choose* h_m *according to Eq.4:*
- *(2) Update* $w_i^{(m)} \leftarrow w_i^{(m)} \exp[-y_i h_m(x_i)]$, *and Normalize to* $\sum_i w_i^{(m)} = 1$;

3.(Output)

$H(x) = Sign[\sum_{m=1}^{M} h_m(x)]$

Figure 1: RealBoost Algorithm.

The "margin" of an example (x, y) achieved by $H(x)$ on the training examples can be defined as $yH(x)$. This can be considered as a measure of the confidence of the h's prediction. The following criterion measures the bound on classification error [11]

$$J(H(x)) = E_w(e^{-yH(x)}) = \sum_i e^{-y_i H(x)} \tag{2}$$

AdaBoost construct $h(x)$ by stage-wise minimization of Eq. 2. Given the current $H_{M-1}(x) = \sum_{m=1}^{M-1} h_m(x)$, the best $h_M(x)$ for the new strong classifier

$$H_M(x) = H_{M-1}(x) + h_M(x) \tag{3}$$

is the one which leads to the minimum cost

$$h_M = arg\min_{h^+} J(H_{M-1}(x) + h^+(x)) \tag{4}$$

Figure 2 shows example CFHIP detection results using AdaBoost algorithm.

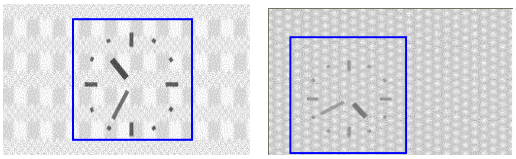


Figure 2: Example CFHIP challenges and detection results using the AdaBoost algorithm

2.2 Clock feature localization and time telling

Once the clock is detected in the image, we next need to estimate the clock center for further analysis. To avoid the effect of clock hands on clock center estimation, the inner part (33%) of the detected window is masked out (Figure 3

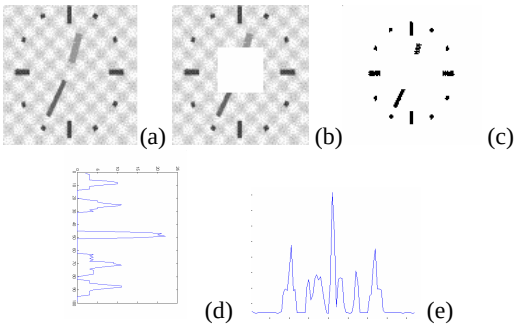
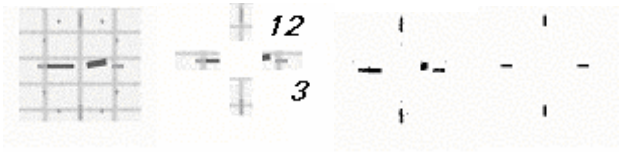


Figure 3: Clock center estimation: (a) Cropped clock region. (b) Masking out inner part of the detected clock region. (c) Binarization result. (d) Projection along vertical direction. (e) Projection along horizontal direction.

(b)). Then the image is binarized using an adaptive threshold, learned from the training samples (Figure 3 (c)). If we build histograms along both horizontal and vertical directions, the clock center has peaks in both histograms (Figure 3 (d) and (e)). The intersection of the peaks gives us the location of the clock center.

Because the clock face is an *ellipse* instead of a *circle*, we also need to locate the clock tick marks. We divide the tick marks into two classes: block shapes and point shapes. We give the block shapes index numbers 3, 6, 9 and 12 and the index numbers for point shapes are 1, 2, 4, 5, 7, 8, 10 and 11. Block shapes 3, 6, 9 and 12 are detected at first. After estimating the center of the clock, the four candidate regions of 3, 6, 9 and 12 are cropped out along the vertical and horizontal directions of the clock center (Figure 4 (b)).



(a) (b) (c) (d)

Figure 4. Detection of 3, 6, 9, 12 clock feature points: (a) Crop the clock region. (b) Crop the clock sub-regions. (c) Binarization result. (d) Detected block shape features.

Because in the candidate regions block shapes are darker than the background and the number of pixels that belong to the block shape features is almost fixed, an adaptive threshold is chosen using Eq.5:

$$\sum_{i=1}^{T} h(i) \le \theta \quad \sum_{i=1}^{T+1} h(i) > \theta \tag{5}$$

where $h(i)$ is the histogram bin i , which represents the number of pixels in the image that have intensity value i . θ is the threshold of the number of pixels which belongs to the block shapes and is learned from training samples. The segmentation result is shown in Figure 4 (c). A rectangle template is then applied in each small region, and the location with maximum correlation with the template is consider as the block shape location (see Figure 4 (d)). Point shapes, e.g., 2, 4, 5, 7, 8, 10 and 11, can be detected in a similar way.

After the previous steps, we already knew the location of the clock tick marks. If we mask out these tick marks, only the two clock hands will be left in the clock region (Figure 5 (a)). An adaptive threshold is again learned from training examples and is used to segment out the clock hands (Figure 5 (b)). A median filter is used to remove noise and the result is shown in Figure 5 (c). So far, we have obtained the clock center, clock tick marks and the clock hands. Lines are drawn between the clock center and the tick marks for time analysis (Figure 5 (d)).

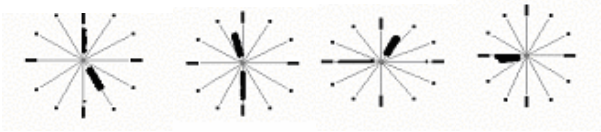
(a) (b) (c) (d)

Figure 5: Hands segmentation and time analysis: (a) Mask out clock tick marks. (b) Segmentation result. (c) Noise removal using median filter. (d) Lines between clock center and tick marks.

The only step left now is to distinguish the minute hand and the hour hand. We observe from the CFHIP training examples that:

(1) The minute hand is always along the direction between the clock center and one of the twelve clock features.
(2) The minute hand is longer but thinner than the hour hand.

Using the above observations, we can find the direction, which has the largest number of black pixels among the twelve directions, as the minute hand direction. Once we find the minute hand direction, it will provide further information for estimating the hour hand. For example, if the minute hand points to clock feature 12 (Figure 6 (a)), then the hour hand must be along one of the twelve lines connecting clock center and tick marks. Similarly, if the minute hand points to clock feature 6 (Figure 6 (b)), then the hour hand must be at the middle of any two adjacent lines. Similar rules can be drawn when the minute hand points to feature 9 (Figure 6 (c) and (d)), and any other features.



(a) (b) (c) (d)

Figure 6: Some clock hands segmentation results.

3. EXPERIMENTAL RESULTS

1,000 CFHIP images are collected from the web [6], 300 for training and 700 for testing. The clock locations of the 300 training images are hand labeled for the Adaboost classifier. The 300 training images generate 300 clock training examples. The non-clock training examples are

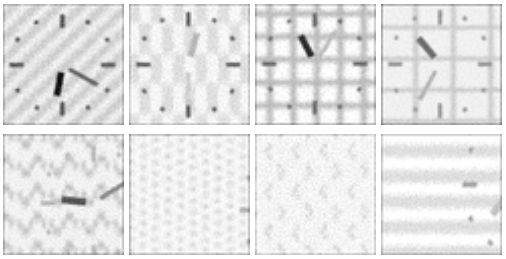
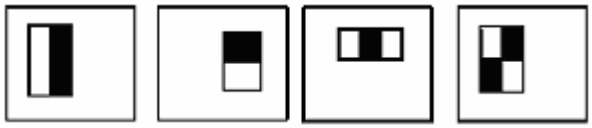


Figure 7: Clock training examples (top row) and non-clock training examples (bottom row).

generated from the background using the bootstrap method [9], and a total of 1,200 non-clock training examples are obtained. Both clock and non-clock training examples are then normalized to 60*60 pixels (see Figure 7). We next report experimental results.

To achieve real-time speed, we use 4 types of Harr-wavelet-like features for clock/non-clock classification (see Figure 8) [12]. We achieve 200 ms for detecting a clock running in Matlab. Given that the base resolution of the detector is 60x60, changing the size and location of the four types of box features will generate 4,626,060 features. If we restrict the height and width of a box to be multiples of 2, we can decrease the number of candidate features to 327,299.



(a) (b) (c) (d)

Figure 8: Four types of features.

Using AdaBoost algorithm, the 40 most important features were selected automatically from the 327,299 candidate features. The clock detector is the linear combination of these selected features. The first two selected features are shown in Figure 9. It is interesting to observe that even though the clocks are of different sizes, shapes and colors, and embedded in different background, Adaboost is able to learn the first two most important features to be the block shapes 3 and 9. If the clock face were an irregular shape instead of an ellipse, it would have been harder to break the CFHIP.

We apply the clock detector on the 700 testing images. For each image, the clock detector scans across the image at multiple scales and locations. Scaling is achieved by scaling the detectors themselves, rather than scaling the image [12]. We scale the detectors using a factor of 1.25. The sub-window with maximum value $H_M(x)$ (Eq.1) is detected as the clock region. Among 700 images, clocks in 632 images are detected correctly. The detection rate is 90.3%. From these 632 corrected detected clocks, our system correctly tells the time for 612 of them. The overall time recognition rate of our system is therefore 612/700 = 87.4%. Figure 10 shows an example of correctly recognized time.

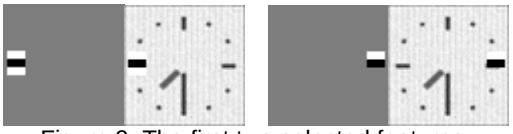


Figure 9: The first two selected features.

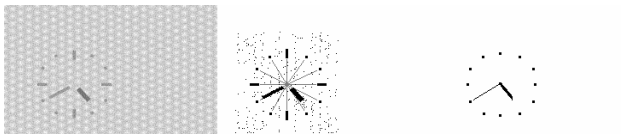
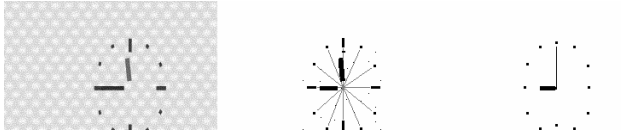


Figure 10: telling the time: (a) Original image. (b) Segmentation result. (c) Synthesized clock based on the time told by the system.



(a) (b) (c)

Figure 11. An example failure case: fail to discriminate hour and minute hand: (a) Original image. (b) Segmentation result. (c) Synthesized clock based on the time told by the system.

There are two types of failures: failure of clock detector and failure of minute/hour hands classification. An example failure case is shown in Figure 11. Our algorithm mistakes the minute hand as the hour hand, and estimates the time to be 9:00 instead of 11:45.

4. CONCLUSIONS

In this paper, we motivated the importance of a new research area called HIP. We first briefly reviewed existing HIPs with focus on the CFHIP, and then presented an algorithm that can defeat CFHIP at 87.4%. We envision the battle between the two sides of HIP will advance the AI research.

5. REFERENCES

[1] Ahn, L., Blum, M., and Hopper, N. J., Telling humans and computers apart (Automatically) or How lazy cryptographers do AI, Technical Report CMU-CS-02-117, February, 2002.

[2] AltaVista's Add URL site: altavista.com/sites/addurl/newurl.

[3] Baird, H.S., and Popat, K., Human Interactive Proofs and Document Image Analysis," Proc., 5th IAPR Workshop on Document Analysis Systems, Princeton, NJ, 2002.

[4] Chew, M. and Baird, H. S., BaffleText: a Human Interactive Proof," Proc., 10th IS&T/SPIE Document Recognition & Retrieval Conf., Santa Clara, CA, January 22, 2003.

[5] Friedman, J., Hastie, T., Tibshirani, R. Additive logistic regression: a statistical view of boosting. Technical report, Department of Stastics, Sequoia Hall, Stanford Univerity (1998).

[6] Izymail, http://izymail.com/imo_getstarted.aspx.

[7] Mori, G. and Malik, J., Recognizing objects in adversarial clutter: breaking a visual CAPTCHA, Proc. of IEEE CVPR, 2003.

[8] Naor, M., Verification of a human in the loop or identification via the Turing test, unpublished notes, September 13, 1996.

[9] Rowley, H., Baluja, S., and Kanade, T., "Neural Network-based face detection". CVPR, 1996.

[10] Rui, Y. and Liu, Z., ARTiFACIAL: Automated Reverse Turing test using FACIAL features, ACM/Springer Multimedia Systems Journal, Spring 2004.

[11] Schapire, R.E., and Singer, Y., Improved boosting algorithms using confidence-rated predictions. Machine Learning, 37(3):297-336, December 1999.

[12] Viola, P., and Jones, M.J., "Robust real-time object detection", Technical Report Series, Compaq Cambridge Research Laboratory, CRL 2001/01, Feb, 2001.

[13] Zhang, Z., MingJing Li, Stan Li and HongJiang Zhang, "Multiview Face Detection with FloatBoost", WACV, 2002.