Using Character Recognition and Segmentation to Tell Computer from Humans

Patrice Y. Simard, Richard Szeliski, Josh Benaloh, Julien Couvreur, and Iulian Calinov
One Microsoft Way, Redmond, WA 98052
{patrice,szeliski,benaloh, jcouv, iulianc}@microsoft.com

Abstract

How do you tell a computer from a human? The situation arises often on the Internet, when online polls are conducted, accounts are requested, undesired email is received, and chat-rooms are spammed. The approach we use is to create a visual challenge that is easy for humans but difficult for a computer. More specifically, our challenge is to recognize a string of random distorted characters. To pass the challenge, the subject must type in the correct corresponding ASCII string. From an OCR point of view, this problem is interesting because our goal is to use the vast amount of accumulated knowledge to defeat the state of the art OCR algorithms. This is a role reversal from traditional OCR research.

Unlike many other systems, our algorithm is based on the assumption that segmentation is much more difficult than recognition. Our image challenges present hard segmentation problems that humans are particularly apt at solving. The technology is currently being used in MSN's Hotmail registration system, where it has significantly reduced daily registration rate with minimal Consumer Support impact.

1. Introduction

Work on distinguishing computers from humans traces back to the original Turing Test [1] which asks that a human distinguish between another human and a machine by asking questions of both. Recent interest has turned to developing systems that allow a computer to distinguish between another computer and a human to enable the construction of automatic filters to prevent automated scripts from utilizing services intended for humans [2]. Such systems have been termed Human Interactive Proofs (HIPs) [3] or Completely Automated Public Turing Tests to Tell Computers and Humans Apart (CAPTCHAs) [4]. An overview of the work in this area can be found in [5].

The CMU CAPTCHA project and the work at PARC [6] are presently the most advanced of these efforts and include various text recognition, visual pattern recognition, image processing, and audio recognition challenges.

Construction of HIPs that are of practical value is difficult because it is not sufficient to develop challenges at which humans are somewhat more successful than machines. This is because there is little cost in having an automatic attacker that fails most of the time. In practice, if one wants to block automated scripts, a challenge at which humans are 99% successful and machines are 1% successful is still not sufficient if the cost of failures and repetitions is low for a machine. Thus, to be useful, a HIP must make the cost of an automated attack high enough to discourage repeated guessing. In a strong sense, a HIP is successful if the cost of answering challenges with a machine is higher than the cost of soliciting humans to perform the same task.

1.1. Synthesis and analysis

One important advantage of the HIP approach is that generating a HIP is a "synthesis" task while breaking it is an "analysis" task. Computationally, Synthesis is usually orders of magnitude easier than analysis, especially when synthesis is lossy. For instance, computing a 2D projection of a 3D object, merging the projection into a larger image, and occluding it with other objects are all trivial image synthesis tasks. Locating and reconstructing the original object is much more difficult and is still an unsolved vision problem.

Fortunately, the task for generating a HIP is usually a synthesis one, while breaking the HIP can be made an analysis task. This generally makes it easier to modify a HIP paradigm than a system that has been constructed to analyze it. Even if analysis techniques catch up with a particular HIP system on one axis, a simple modification can often create a new design that is beyond the reach of analysis tools on some other axis. This has the potential to place an enormous burden on adversaries who seek to defeat a HIP system. For instance, if a HIP required counting how many instances of chairs appear in an image, and if an adversarial party had painstakingly built a chair detector, the HIP paradigm could easily change to counting butterflies, and the adversary might have to build a butterfly detector from scratch. This is an arms race stacked in favor of HIPs.

It is interesting to note that other races can be stacked differently. For instance SPAM filtering put the burden of analysis on the filtering while the spammer has the much easier task of synthesis. This is also an arm's race, but it is stacked in the favor of the spammer which does the synthesis. When the filter blocks the spam, the spammer can very cheaply generate different forms of email. Because analysis is so much harder than synthesis, it may eventually become difficult even for the human to sort the SPAM from genuine email based on content alone. An interesting strategy is to turn the tables around and challenge the spammers with a HIP.

1.2. Character recognition versus other visual HIPs

There are many synthesis tasks which might be appropriate as HIPs. Among visual challenges, we have chosen character recognition for several reasons. First, the task must be extremely simple and universal. Humans may have to solve lots of HIPs in different situations. If they have to read and understand different instructions for each HIP, it puts an important cognitive burden on the subject who is likely to eventually refuse to do it. For instance, recognizing random objects in a scene may be a good challenge, but it requires instructions as to which objects must be recognized, and a description of the dictionary of the object names. Universality is unlikely to happen as different and competing entities are likely to generate HIPs with incompatible instructions. Counting objects in a scene also requires instructions and potentially large images since we need to count several different entities to maintain a less than one in a thousand chance of an adversary being successful by random guessing. This then also becomes a fairly complex cognitive task for human subjects.

Identifying letters, on the other hand, has several advantages. The dictionary is obvious and each class has an assigned button on the keyboard. Letters/digits have less ambiguity and are more language independent than other object names, and every computer has primitives to draw them on a bitmap. Optical character recognition (OCR) is a very well understood problem for both printed and cursive text, and we know the strengths, weaknesses, and complexity of the state of the art algorithms [7]. Arguably, if any visual HIP has a chance of becoming universal, character recognition HIPs are probably it.

1.3. Pure classification tasks

Pure classification tasks, however, can be solved in a fairly automatic way. Since the ‘P’ in “CAPTCHA” stands for public, it is possible to generate a large database of challenges with their labels. We can then train state of the art learning algorithms, such as neural networks or Support Vector Machines (SVM), on this database and expect fairly good results. Since this procedure is automatic, every time the HIP paradigm changes, a new database can be generated and fed to the learning algorithm to break the new paradigm. A large number of analysis problems are solved is such a way. Indeed, many in the field consider that at least for printed characters, OCR is a solved problem. Even for cursive and highly distorted characters, the state of the art OCRs are remarkably robust and can recognize letters under extreme conditions of distortion and noise [7].

We therefore propose to augment the classification task with a segmentation task. In section 2, we argue that segmentation is intrinsically harder than classification and still presents an insurmountable challenge for the current state of the art algorithms.

This is not equivalent to saying that the problem will never be solved. Clearly, if humans can do it, we have no way to assert that machines cannot. However, humans benefit from massive parallel visual processing and yet-to-be-understood multi-scale reasoning algorithms. When computer algorithms approach human segmentation abilities, many important vision tasks will be resolved as well. In the meantime, segmentation HIPs can provide useful services.

2. Recognition versus segmentation

Is pattern recognition fundamentally different from pattern segmentation and detection? Can segmentation be cast as a recognition problem?

2.1. The “everything else” space

One might argue that it is possible to build a segmenter out of a recognizer by running the recognizer over the entire image, and training it to only fire when valid patterns are encountered. Despite the fact that this approach may not be practical at test time (for computational reasons), the main problem is that training such a recognizer might be prohibitively expensive. The recognizer needs to distinguish valid characters from bad input. This is potentially very difficult because the class boundary between a given class and everything else is typically much more complex than the class boundary between two classes, which is often linear for all practical purposes. The intrinsic dimensionality of the everything-else space can be much larger than the intrinsic dimensionality of the space generated by valid examples. Depending on the problem, the space of “sampled” everything-else can be zero (every sample is a valid pattern), large (e.g. mis-aligned or cropped valid patterns) or hopelessly huge (purposefully designed “garbage” patterns). Therefore, to train a classifier to recognize everything-else can be a much harder task than regular classification between given classes.

2.2. Segmentation successes

There are a few cases where classifiers are trained for detection and segmentation. For instance neural networks have been successfully used for the segmentation of cursive handwriting [8]. However, this success relies on three contributing factors. The cursive segmentation problem is 1-dimentional, the everything-else space is limited to transition from valid classes to valid classes (mis-alignment), and classification is helped by a language model. If the segmentation was 2 dimensional, the dynamic programming algorithm provided by Viterbi would not work and would be prohibitively expensive [9]. Because the everything-else space is relatively small, it can be sampled and learned. Even so, the performance would not be satisfactory if a language model was not used to filter out mis-classification of "everything-else as valid characters.

Another example is the use of a boosting classifier for face detection [10]. In this case, boosting is used to select features to detect faces in an image, with a remarkably low rate of false positives. The success of human face detection is due to the presence of very characteristic features around faces, e.g. darker regions around the eyes, nose and mouth, with a fixed relationship between them, which can be detected by a clever algorithm, and used for localization. If a set of low level features can be linked with the presence of an object, localization is a much easier task. Of course, in an adversarial case, extra care is taken to ensure that valid and invalid patterns created by distortions have similar features.

2.3. Making segmentation difficult

Unlike cursive handwriting and face detection where the segmentation task is fixed, we have the freedom to set our problem to make segmentation difficult. In particular, we can choose a 2D setting, generate a very high dimensional space for everything-else, and remove many features that could distinguish valid characters from random patterns. We also do not limit our selection of strings to those that appear in a dictionary. Figure 1 is a simple illustration of the difference between classification and segmentation.

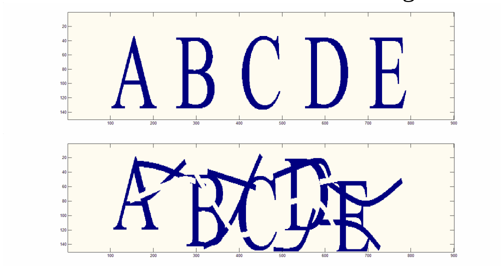

Figure 1. Top: No distortions. Bottom: Random placements and arcs drawn with foreground (2 per letter) and background (1 per letter) colors.

The top portion of the figure contains the unaltered letters 'ABCDE'. The same letters at the bottom have been randomly moved and covered with arcs of both foreground and background color. Even with such simple processing, off the shelf OCRs fail to recognize these letters. For instance, when the Scansoft OCR was fed the tiff images from figure 2,

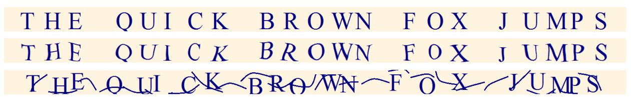

Figure 2. Top: No distortions. Middle: image warping. Bottom: Random positions, and 1 foreground arc per letter.

it produced respectively "THE QUICK BROWN FOX JUMPS", "TIlE QUICK BkOWN FOX iUMPS", and "YaJ _oJJI – YMMP". Distortions such as warping (which will be described in the next section) clearly affect recognition but not nearly as much as arcs and random positioning. One reason is that letter placement is not known a-priori in the last example and random positions yield fictitious characters that are as valid to the OCR as the true characters. Yet this task remains fairly easy for humans.

It should be noted that in this example, a clever programmer could easily distinguish the added strokes from the letters by looking for constant thickness, horizontal and vertical alignments, serifs, etc. and solve this problem. The algorithm, however, would have to be custom for this particular alteration. We will show in later sections much more challenging images that even intensive custom programming might not break.

3. Distortions

The simplest distortions are random rotations and translations of the characters, but most good OCR software is insensitive to such distortions. We therefore add a general warping function. First, we compute a pair of warp fields by generating random (white) noise and then use a separable recursive low-pass filter [11]. Two parameters control this step: the cut-off frequency of the low-pass filter and a scaling factor that multiplies the whole (normalized) field. The first parameter controls the smoothness of the deformation while the second parameter controls its intensity. These warp fields are then used to re-sample the originally rendered character bitmap using an inverse warping algorithm [12].

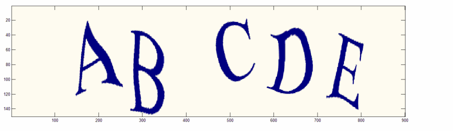

Figure 3. Image after applying warp field.

Figure 3 shows the resulting image after the original image has been deformed with a low-frequency random warp field.

4. Arcs

In order to make the character segmentation task more difficult, we draw random arcs over the generated letters. Arcs are drawn in both the foreground and background color, in order to potentially link or break apart letter features.

Each arc is drawn as a Cardinal spline [13] with three control points. The end points of each arc are chosen (with random perturbations) to roughly bridge the space between adjacent characters. The midpoint is chosen as a random deflection of the average position of the endpoints, in order to give each arc some random curvature. Figure 4 shows the resulting image after some random arcs have been superimposed.

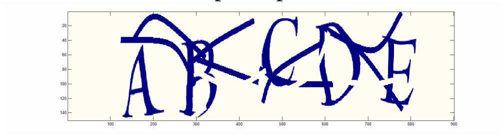

Figure 4. Image after adding random arcs.

5. Thickness Variations

Arcs and letters might be distinguishable because of their thickness. If this was the case, an effective adversarial attack could be to identify arcs and remove them, or identify the corresponding pixels as "don't care". Once the letters are identified, segmentation and therefore classification becomes much easier. (We believe that warping alone does not prevent breaking the challenges.) It is therefore imperative to make the segmentation of the added arcs from the letters extremely difficult.

5.1. Morphological transformations

Our first attempt was to use morphological transformations, such as dilation and erosion on the characters and arcs to vary their thickness. To do this, compute the distance transform [14] on the binary image, and set the pixels to foreground if the transform distance is less than a random value taken from a smoothed random scalar field (obtained using a similar algorithm as for warping). We then do the same on the negative images for the erosion. To prevent characters from disappearing in places, we also estimate the zero partial derivatives of the distance transform (which correspond to medial axes) and prevent the corresponding pixels from being changed by the erosion and dilation. An example of the resulting image is shown in Figure 5.

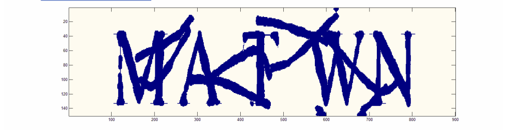

Figure 5. Thickness variations from dilations and erosions.

Unfortunately, this approach is not visually appealing because it makes the characters appear to be dirty. The erosion can also sometimes eat too much of a character (e.g. left leg of first 'M' or top of 'T' in the figure). We only implemented dilation for binary images, which has the drawback of loosing aliasing information on the letters (which is also less appealing). Oversampling or implementing dilation for gray level images makes this method computationally expensive.

5.2. Local warping

A better alternative to dilation is to subject the whole image to a local warping. This step is similar to the previous warping step described in the distortion section. The difference is that the cut-off frequency of the low pass filter is higher, while the intensity is lower. The result is that warping is more localized and gives the visual appearance of changes in thickness. This process makes distinguishing arcs from pieces of letters more difficult but is still esthetically pleasing and computationally cheap. The result of local warping on the image and arcs (with no other distortion) is shown in Figure 6.

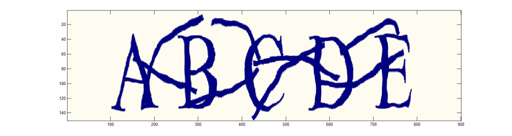

Figure 6. Thickness variations from local warping.

6. Deployment

Many combinations of alterations can be used to make the challenge more difficult to computers. We can increase the size of the image (make it truly 2D), the number of arcs, the number of letters, and even make the challenge a "find k out of n" at very high difficulty settings.

Using the above methods, a single distortion parameter can be introduced to adjust the complexity of the image. Initially, the setting could be a low or no distortion. When the adversarial party starts breaking the challenges, the knob can be turned up to make the challenges harder. Figure 7 shows, for example, the effect of various settings ranging from no distortion to high distortion.

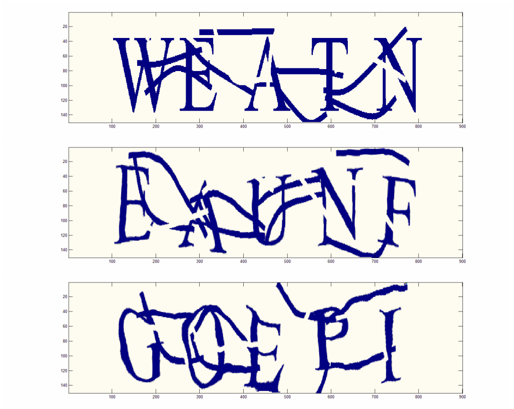

Figure 7. Images with increasing amount of distortions.

Our HIP was deployed in MSN's hotmail registration system (www.hotmail.com) on December 12th 2002. A 19% drop in daily registration was immediately observed. At the time of this publication, the Customer Support inquiries related to this implementation have been minimal, indicating that the HIP has been readily accepted by hotmail users.

7. Conclusion

Effective Human Interactive Proof systems are becoming an important component for many on-line services. By making the cost prohibitive for a machine to gain access without the expenditure of human capital, one can protect many services from unwanted automated intrusions. Gating services such as electronic mail, admission to chat rooms, and on-line polling can help preserve the integrity of these services.

The techniques described in this paper leverage the human advantage over machines at segmenting text into its constituent characters. By exploiting these advantages, text-based challenges can be generated that are fairly easily read by humans but are well beyond the current capabilities of automated text recognizers.

Challenges can be generated efficiently, and their difficulty can be easily adjusted in response to user feedback or technological advances.

While we believe that these methods provide a high degree of assurance, further work is likely to be necessary to stay ahead of developments in hardware and software that may better mimic the parallel processes utilized by humans for text recognition.

References

[1] A. M. Turing, *Computing Machinery and Intelligence,* Mind, vol. 59, no. 236, pp. 433-460. 1950.

[2] L. von Ahn, M. Blum, and J. Langford, "Telling Computers and Humans Apart (Automatically) or How Lazy Cryptographers do AI". To appear in Communications of the ACM.

[3] First Workshop on Human Interactive Proofs, Palo Alto, CA, January 2002.

[4] L. von Ahn, M. Blum, and J. Langford, *The Captcha Project.* http://www.captcha.net.

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

[6] H. S. Baird, A. L. Coates, R. Fateman, "PessimalPrint: a Reverse Turing Test", *Int'l J. on Document Analysis and Recognition*, vol. 5, pp-158-163, 2003.

[7] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-Based Learning Applied to Document Recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, Nov. 1998.

[8] Y. Bengio, Y. LeCun, C. Nohl, and C. Burges, "LeRec: A NN/HMM Hybrid for On-Line Handwriting Recognition," *Neural Computation*, vol. 7, no. 6, pp. 1289--1303, 1995.

[9] E. Levin and R. Pieraccini. *Dynamic planar warping for optical character recognition*. Proceedings of ICASSP'92, III:149-152, 1992.

[10] P. Viola and M. Jones. *Rapid object detection using a boosted cascade of simple features*. In Proc. Computer Vision and Pattern Recognition, I:511-518, Kauai, 2001.

[11] R. Deriche, "Fast Algorithms for Low-Level Vision", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1), January 1990, pp. 78-87.

[12] Wolberg, G., *Digital Image Warping*, IEEE Computer Society Press, Los Alamitos, 1990.

[13] Bartels, R. H., Beatty, J. C., and Barsky, B. A., *An Introduction to Splines for use in Computer Graphics and Geeometric Modeling,* Morgan Kaufmann Publishers, Los Altos, 1987.

[14] P. Danielsson. Euclidean distance mapping. *Computer Graphics and Image Processing*, 14:227--248, 1980.