Distortion Estimation Techniques in Solving Visual CAPTCHAs

Gabriel Moy, Nathan Jones, Curt Harkless, and Randall Potter
Areté Associates
Sherman Oaks, CA 91403
Email: {moy,jones,harkless,potter}@arete.com

Abstract— **This paper describes two distortion estimation techniques for object recognition that solve EZ-Gimpy and Gimpy-r, two of the visual CAPTCHAs ("Completely Automated Public Turing test to tell Computers and Humans Apart") with high degrees of success. A CAPTCHA is a program that generates and grades tests that most humans can pass but current computer programs cannot pass. We have developed a correlation algorithm that correctly identifies the word in an EZ-Gimpy challenge image 99% of the time and a direct distortion estimation algorithm that correctly identifies the four letters in a Gimpy-r challenge image 78% of the time.**

I. Introduction

Many computer vision applications rely on accurate object recognition for success. While there is no unifying object recognition technique, it is important to advance strategies to take care of specific problems such as accounting for noise from the addition of background clutter and distortions. Visual CAPTCHAs like EZ-Gimpy (Fig. 1) and Gimpy-r (Fig. 2) are good examples of simple objects with background clutter and distortions. Manuel Blum's group at Carnegie Mellon University describes different CAPTCHAs based on visual or audio information at `http://www.captcha.net` [1].

The visual CAPTCHAs include the Gimpy family of tests, Bongo, and Pix. Gimpy involves identifying three of approximately seven distinct words in an image. EZ-Gimpy is a simpler version that only uses one word, while Gimpy-r has four random letters. The audio CAPTCHA, Sounds, is an audio version of Gimpy. A sequence of letters or words is rendered, distorted, then played. The test is to determine the contents of the sound clip.

In both EZ-Gimpy and Gimpy-r, the user is presented with a 290 pixel $\times$ 80 pixel JPEG image and prompted to enter a "guess" as to what word or sequence of letters is shown. The letters in EZ-Gimpy are from one font, and the background clutter can consist of white noise, a grid, or other patterns such as swirls. The letters in Gimpy-r are from two fonts, and the background clutter is mostly different colored distortion patterns such as boxes, waves, and ripples.

Visual CAPTCHAs are used to prevent spammers from performing automated techniques in acquiring free email accounts from sites such as Yahoo and to stop automated ticket purchases from Ticketmaster. Furthermore, any program that passes the tests generated by a CAPTCHA can be used to solve a hard unsolved AI problem [2]. We take on the problem of EZ-Gimpy and Gimpy-r not only to show that they are

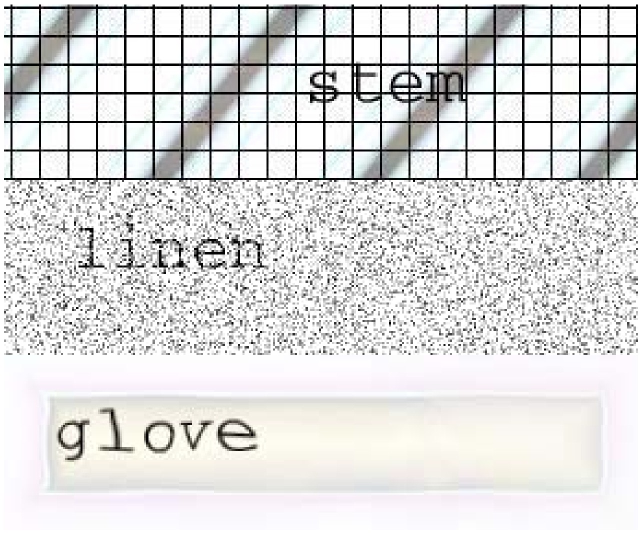

Fig. 1. Three EZ-Gimpy challenge images.

ineffective deterrents but also to advance the progress on AI problems.

The EZ-Gimpy test uses a dictionary of 561 words while the Gimpy-r test uses a set of four random letters from a dictionary of 19 letters. The approaches we use to solve EZ-Gimpy at a 99% level and Gimpy-r at a 78% level add to our collection of object recognition tools. The strategies consider the specific problems of cluttered backgrounds and distorted letters, but do not take into account other issues such as sight angle, lighting effects, context, and camouflage. In the case of EZ-Gimpy, we use a whole object recognition approach against each object in the dictionary, since the dictionary is relatively small. Our approach is very different from the bigram approach of Mori and Malik [4] or the chamfer matching approach of Thayananthan, et. al, [7] each of which achieve a 93% success rate. With Gimpy-r, the dictionary has $19^4 = 130,321$ entries. A comparison to each entry would be too time consuming, so we break down the problem into four individual letter recognition problems. Neither the holistic nor individual letter approach to word recognition is new. Madhvanath and Govindaraju [3] have used a holistic approach in handwriting recognition, while Plamondon and Srihari [5] present a survey of holistic and segmented approaches for handwriting recognition.

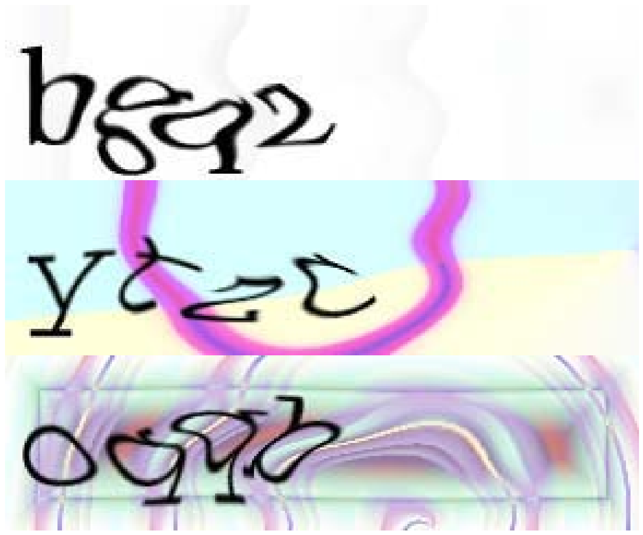

Fig. 2. Three Gimpy-r challenge image.

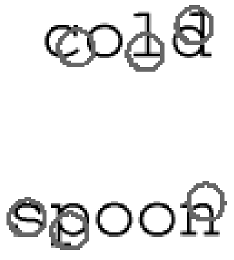

Fig. 3. The three cores of two template images.

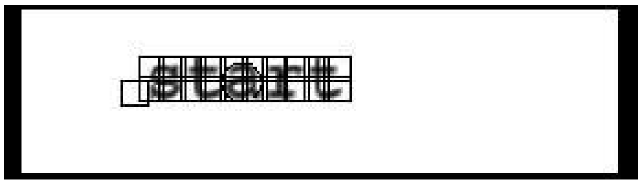

Fig. 4. A template image split into 24 minipatches.

Section II describes our correlation algorithm for solving EZ-Gimpy. Section III describes our distortion estimation algorithm for solving Gimpy-r. We discuss our conclusions in Section IV.

II. Matching whole objects by correlation

Given a small set of template images, such as those in EZ-Gimpy, we are able to test the challenge image against each of the template images. Instead of trying to deduce which individual letters are in the challenge image, we find the best correlated template image. Our distortion estimation with correlation approach uses what we call a “core” and “minipatch” framework. In each template image, we identify 3 cores and 24 minipatches. We use variations of the cores and minipatches to estimate distortions and then find which distorted template image best correlates to the challenge image.

A. Cores

We define a core as an area that is most distinct, i.e., least correlated with the rest of the image. Given an image that is 290 pixels $\times$ 80 pixels, we choose a circular core with a 16 pixel diameter. We compare each possible 16 pixel disc with every other 16 pixel disc and find the three least correlated sections that do not overlap. These cores represent the most distinctive features of the word (Fig. 3) and will be the anchor points between the template and challenge image.

B. Minipatches

We next split the template word into 24 small overlapping sections, which we call minipatches (Fig. 4). Starting from the original minipatch, we create five rotated versions. With the five rotations, we independently shrink or stretch the minipatch in the X and Y directions, giving 45 variations per minipatch. We also keep track of the core positions with respect to the minipatch positions. The variations on the minipatches represent the types of distortions encountered. If larger ranges of distortions need to be estimated, we would use a larger set of minipatches with more variations, with the side effect of increased execution time.

C. Matching

Given a challenge image, we go through multiple steps to arrive at a metric of correlation to the template images. The steps are as follows:

- Background removal
- Template core anchoring onto the challenge image
- Optimal minipatch placement and correlation calculation

We now discuss each of these steps in detail.

1) Background removal: Given the nature of background clutter, we are able to remove the background to a great extent without losing the important information, in this case, the challenge word. Instead of using the intensity ($\sqrt{red^2 + green^2 + blue^2}$) of each pixel, we use the maximum color component ($max(red, green, blue)$). We then detect and remove the grid if one exists by looking for a periodic sequence of lines. Next, we apply a threshold to the image, keeping all pixels greater than the threshold. We determine if the image is noisy by looking at the intensity variance of 64 pixel $\times$ 32 pixel sections located at the corners of the image. If the image is noisy, we apply a thresholding technique that takes into account the number of neighbors that are also above the threshold. After this processing, the image includes the challenge word and small bits of background noise.

2) Core anchoring: With the clean challenge image, we test it against the cores and minipatches of each template image. Starting with the three cores, the best correlated locations are found (Figs. 5 and 6). We use the original core as well as independent X and Y stretching or shrinking of the core, giving nine variations of the core for anchoring. We only use

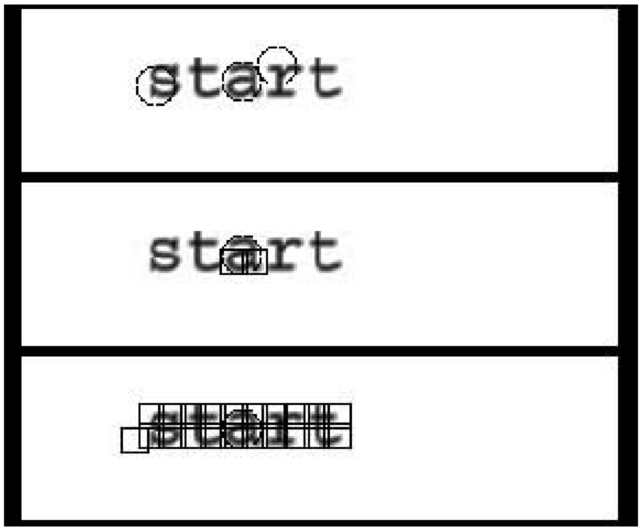

Fig. 5. An easy EZ-Gimpy challenge image showing cores, core anchoring with two minipatches, and core anchoring with all twenty-four minipatches. The correct template image is used.

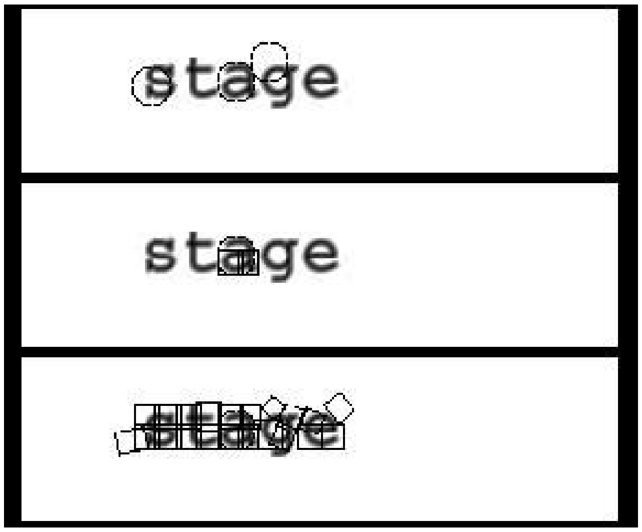

Fig. 6. An easy EZ-Gimpy challenge image showing cores, core anchoring with two minipatches, and core anchoring with all twenty-four minipatches. An incorrect template image is used.

nine variations due to execution time issues. Since we know the relative position of the core to the minipatches, we can then lay out the set of minipatches.

3) Minipatch placement and correlation calculation: Starting with the best correlated core, we first choose the minipatch closest to the core location. We compare the 45 different variations to the core area and find the best correlated minipatch with respect to position, rotation, and X and Y stretch or shrink. This information and the relative location of the next minipatch are used to lay out the remaining minipatches, starting from the minipatches closest to the core and working outwards. We repeat the same procedure for the other two cores (Fig. 7). We calculate a correlation for each template, and the highest correlation across all templates is returned as

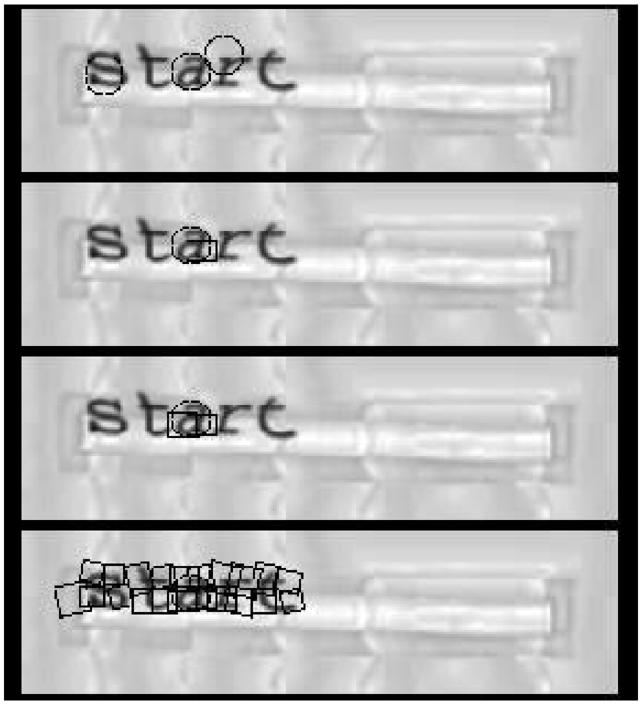

Fig. 7. A typical EZ-Gimpy challenge image with an overlay of correct template cores, and minipatch layouts with one, two, and all twenty-four minipatches.

the "guess".

D. Results

Using a dual 2.0GHz Pentium system, the algortihm, implemented in C, processes each challenge image and returns a guess in approximately 18 seconds. We used a set of 1000 images for development and another set of 1000 images for testing. There are 15 errors in the training set and 10 errors in the test set giving a success rate of 99%. The jump in accuracy is from favorable statistical noise. The core and minipatch approach is a general approach and not limited to the EZ-Gimpy problem. We have also applied it to the fingerprint matching problem with great success.

III. Matching sub-objects by distortion estimation

The core and minipatch approach compares each template image with the challenge image. In the case where there is a very large dictionary and large local distortions, as is the case with Gimpy-r, the computation time becomes unmanageable. There are too many templates and minipatch variations to consider before attaining a guess.

Instead, we use a new multi-stage approach that estimates the local distortions and then finds the template image with the lowest average distortion from the challenge image. From the initial guess, we build confusion matrices in which we apply a maximum likelihood test to maximize success. Geometric constraints are exploited to minimize the search space as well as to differentiate between very similar images.

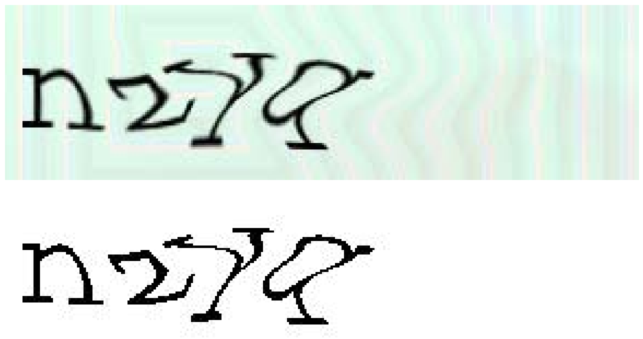

Fig. 8. A Gimpy-r challenge image with the background removed.

In the case of Gimpy-r, the challenge image consists of four sub-images, or letters, chosen randomly from a set of 19 possibilities, giving a dictionary of $19^4 = 130,321$ entries. We split the problem into four independent problems each with a dictionary size of 19. To achieve an 80% success rate for the set of four letters, we need to have a 95% success rate on each individual letter.

We discuss each aspect of the multi-stage algorithm below.

A. Mesh generation and global distortion removal

After applying a similar background removal algorithm as in the core and minipatch approach, we have an image that contains four letters under a moderate continuous distortion (Fig. 8). To estimate the distortion in the area of the letters, we first slice each letter into eight horizontal slices by creating an 18 node (2 wide $\times$ 9 high) mesh whose right and left borders follow a path that is equidistant from the neighboring letters (Fig. 9). The equidistant measure is measured with respect to the whole letter and not just along the horizontal axis. The top and bottom borders are horizontal lines at the maximum and minimum height of the letter, respectively.

Special cases occur at the leftmost and rightmost edges. The left half of the first letter is undistorted, so the left mesh border for the first letter is a straight line. For the rightmost edge, we place an imaginary vertical bar a few pixels away from the end of the fourth letter as the "fifth" letter and use the equidistant path from the fourth and phantom fifth letter as the mesh border.

Stretching the borders to a rectangle circumscribed by the widest nodes and linearly interpolating the letter removes most of the distortion. The mesh generation and stretching are repeated for each of the letters (Fig. 10). This method of border generation and distortion removal uses the ideal assumption that the letter shapes on either side of the border are mirror images of each other.

B. Local distortion estimation

Under the application of a continuous distortion, the topological features of loops and intersections remain loops and intersections. Using this feature, we split our dictionary into letters that contain a loop (`'a'`, `'b'`, `'d'`, `'e'`, `'g'`, `'o'`, `'p'`, `'q'`)

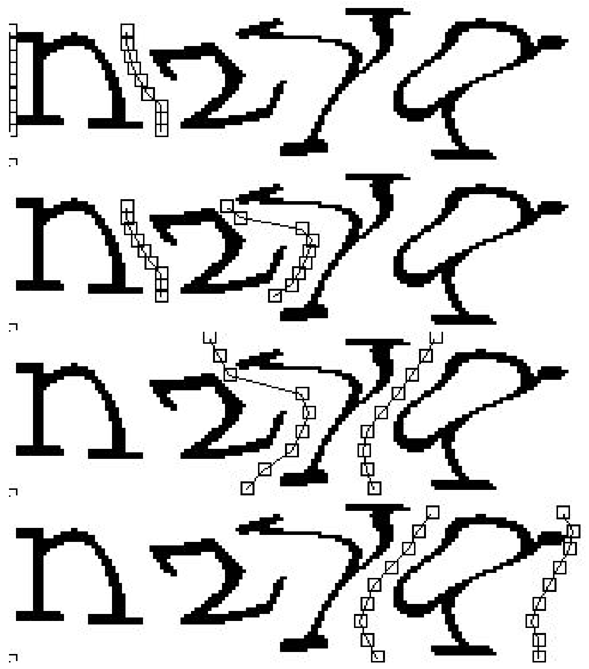

Fig. 9. The mesh nodes for each letter in a Gimpy-r challenge image.

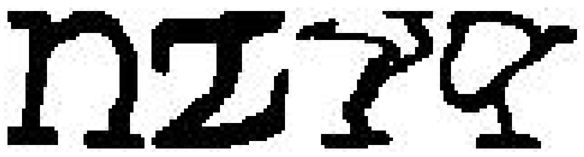

Fig. 10. An undistorted and rescaled gimpy-r challenge image.

and letters that do not contain a loop (`'c'`, `'f'`, `'h'`, `'m'`, `'n'`, `'r'`, `'s'`, `'t'`, `'w'`, `'y'`, `'z'`).

After rescaling the template letters and challenge letter to 40 pixel $\times$ 40 pixel images, we compare the challenge letter against the appropriate sub-dictionary and find the minimum distortion necessary to get from the template image to the challenge image (Figs. 11 and 12). To calculate the distortion, we obtain an ordered list of border points for both the challenge letter and template letter and subsequently find the matching that minimizes border differences. The list of border points are the x and y coordinates of points along the border of the letter. Thus, we let the template and challenge letter border points be denoted as $t = \mathcal{R}^{m\times 2}$ and $c = \mathcal{R}^{n\times 2}, respectively.$ The distortion metric for a specific template and starting point is defined as

$$\min_{l\epsilon[0,m-1]} \frac{1}{m} \sum_{k=0}^{m-1} \| t((k+l)\ MOD\ m) - c(k * \frac{n}{m}) \|_2$$

where $c(k * n/m)$ is the linear interpolation between points $c(\lfloor (k*\frac{n}{m}) \rfloor)$ and $c(\lceil (k*\frac{n}{m}) \rceil)$, and l defines the starting point of the template border points. In other words, the distortion metric is the average magnitude of the vectors needed to change the template letter border to the challenge letter border.

The template with minimum distortion is our guess of which

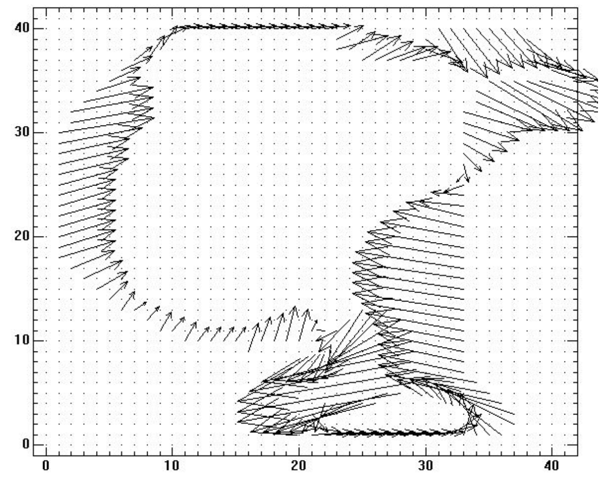

Fig. 11. Distortion needed to go from the template 'q' to challenge 'q'.

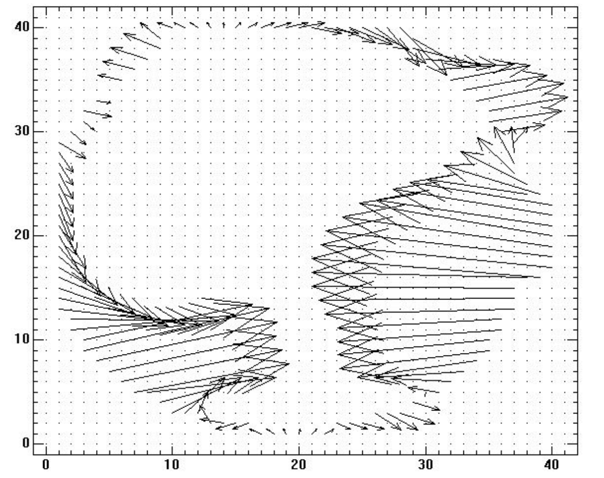

Fig. 12. Distortion needed to go from the template 'o' to challenge 'q'.

letter the challenge sub-image represents. While this technique only yields a 52% success rate on the training database of 564 challenge images, we can use the distortion information to build confusion matrices (Table I). Confusion matrices tabulate the "guess" versus the "truth" to obtain a better understanding of how the algorithm fails.

The confusion matrix shows that the guess can be grouped into an even smaller subset of letters containing the correct letter and a handful of incorrect letters. For example, if the guess is an 'o', then the smaller subset is 'a', 'b', 'd', and 'o'. Curiously, the confusion matrices are non-symmetric, i.e., there are significantly more guesses of a 'g' when the truth is 'q' than vice versa. After distortion estimation on the 564 image test database, of which 553 are successfully parsed, the algorithm successfully identifies all four letters in 294 of the cases and successfully identifies 1883 out of the 2212 individual letters.

Based on the confusion matrix, we apply a maximum

TABLE I

CONFUSION MATRICES

Truth

		a	b	d	e	g	o	p	q
	a			5					
G	b	3		5					
u	d	1							
e	e		3			1		6	
s	g		2	1				2	80
s	o	1	8	11					
	p		1						
	q					4			

Truth

		c	f	h	m	n	r	s	t	w	y	z
	c								2			
	f	1		2						4		
	h		1		1	2				1		
G	m		1							1		
u	n			16	1							
e	r		1	5				2	44	8	4	
s	s		1				5		1	1	3	
s	t	30	4					2		1		1
	w						1	2			1	
	y							1		6		1
	z						17		16			

likelihood rule and exploit geometric properties to finalize the guess.

C. *Maximum likelihood*

With the confusion matrix, the initial guess is now an excellent starting point to decide which letter should be our final guess. Using the distortion metrics described above, we perform a maximum likelihood test to improve the success rate. We consider the difference and magnitude of the distortion metrics between the incorrect guess and the truth. We maximize the number of correct guesses for each pair of looped or non-looped letters by testing over two parameters, α and β, where α is the minimum distortion of the guess and β is the maximum difference between distortions of the guess and truth. For example, we look at the pair 'e' and 'p'. Searching over the space of α and β, we find that if $\alpha > 0.5$ and $\beta < 0.12$, then the guess of 'e' should be changed to a guess of 'p'. This change would yield 6 more correct answers while introducing zero wrong answers. Typical α values are near 0.5 while correct guesses with little distortion have α values near 0.2. This big difference in distortions tells us that there was no good guess and the best of the poor guesses was taken. The correct guess is not too far off, so the number of correct guesses is maximized with an added bias.

D. *Geometric properties*

Many of the pairs of letters in the confusion matrix look somewhat alike, such as ('h','n'), ('g','q'), ('b','o'), and ('c','t'). There are some that do not look at all alike, such as ('f','w') and ('b','g'). For the pair ('h','n'), we compare the height of the arch to the height of the whole letter. For the pair ('b','o'), we look at the residual after removing the area surrounding the looped area. For ('g','q'), the only difference

TABLE II

FINAL CONFUSION MATRICES

		Truth							
		a	b	d	e	g	o	p	q
	a								
G	b			2		1			
u	d	2	2						
e	e								
s	g			1					8
s	o	1							
	p								
	q					1		2	

		Truth										
		c	f	h	m	n	r	s	t	w	y	z
	c			1				1	1			
	f			1					1			
	h		1			3						
G	m											
u	n			1	1							
e	r		1	2					6		2	2
s	s											
s	t	2					14	3			2	2
	w			3			1				2	
	y							1		1		1
	z											

is the small tail at the lower right hand corner of the letter, so we apply a thinning algorithm [6] and look for the branching in the lower right corner. For (‘c’,‘t’), we look for a branching in the top half of the letter. We use many of these types of geometric properties to distinguish pairs of letters to maximize the number of correct guesses.

Even though there are two fonts used in the challenge images, we only use one font for our template images. The major differences in the fonts are ‘g’ versus ‘g’ and that one font is thicker than the other. Since we calculate the average amount of distortion it takes to get the border of a template image to match the border of the challenge image, the thickness property of letters affects all the template image distortion metrics fairly uniformly. For the ‘g’ versus ‘g’ ambiguity, if we find two loops in the letter, then we guess ‘g’.

E. Results

We run our multi-stage algorithm on a 2.4GHz PC as an IDL program. It takes an average of seven seconds to analyze a challenge image and return a guess. After applying our second stage to the 564 image test database, we successfully identified all four letters in 479 of 553 images parsed and 2131 of 2212 individual letters for a success rate of 85%. The image parser had problems when the background could not be removed well enough to identify four distinct letters. The final confusion matrices are shown in Table II.

In a blind test of 736 images obtained from Luis von Ahn of CAPTCHA.net, our algorithm correctly identified 576 images for a success rate of 78%. The discrepancy between our test results and the blind test is due to over-tuning of the maximum likelihood test.

IV. CONCLUSIONS

We have shown two different distortion estimation techniques for identifying objects in clutter. The correlation approach works well when there are small distortions with a small dictionary. Our algorithm has a 99% success rate on EZ-Gimpy. Anchoring the most distinct part of an image and working outwards to see how well the pieces fit is an excellent approach when working with images containing only small distortions. For more accuracy, there is a tradeoff between using more core and minipatch variations versus runtime. For the EZ-Gimpy problem, we chose to use no rotations with three variations in each of the X and Y directions for the cores and five rotations with three variations in each of the X and Y directions for the minipatches to achieve an adequate success rate and execution time.

The direct distortion estimation approach works well on images with large continuous distortions. With the 564 image Gimpy-r training database, our algorithm had an 85% success rate when used with geometric properties and a maximum likelihood test. On the 736 image blind test database, our success rate dropped slightly to 78%. With larger distortions, we estimate the distortion from the image itself, such as the path equidistant between letters in the case of Gimpy-r, to obtain an image closer to the template images.

The distortion estimation algorithms have room for improvement, and they are just proof-of-concepts. There are many optimizations and features available to exploit, such as global distortion information, other sub-object information when compiling current sub-object information, or the optimal number of slices to take per sub-object. As a test, we used 20 slices per sub-object and the baseline distortion estimation accuracy goes up to 54% while runtime increased to 12 seconds per challenge image.

These approaches are not unique to the Gimpy family of problems and are specific applications of a more general toolbox for object recognition.

REFERENCES

[1] L. von Ahn, M. Blum, N. J. Hopper, and J. Langford. The CAPTCHA Web Page: `http://www.captcha.net`. 2000.

[2] L. von Ahn, M. Blum, N. J. Hopper, and J. Langford. CAPTCHA: Using Hard AI Problems for Security. Advances in Cryptology: Eurocrypt 2003, May 2003.

[3] S. Madhvanath and V. Govindaraju. The role of holistic paradigms in handwritten word recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 23(2):149-164, February 2001.

[4] G. Mori and J. Malik. Recognizing objects in adversarial clutter – breaking a visual captcha. Proceedings of the Conference on Computer Vision and Pattern Recognition, vol. 1, Madison, USA, pp. 134-141, June 2003.

[5] R. Plamondon and S. N. Srihari. On-line and off-line handwriting recognition: A comprehensive survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(1):63-84, January 2000.

[6] F.W.M. Stentiford and R.G. Mortimer. Some New Heuristics for Thinning Binary Handprinted Characters for OCR. IEEE Transactions on Systems, Man, and Cybernetics, 13(1):81-84, January/February 1983.

[7] A. Thayananthan, B. Stenger, P. H. S. Torr, and R. Cipolla. Shape Context and Chamfer Matching in Cluttered Scenes. Proceedings of the Conference on Computer Vision and Pattern Recognition, vol. 1, Madison, USA, pp. 127-133, June 2003.