Automated Logo Blurring Using a SIFT-based Approach

Gary Fibiger, Jinfeng Guo, Geerten Klarenbeek, Joost Linthorst, and Erwin Westra

***Abstract*—In films, photos and other visual media, logos of all kinds are commonly present. In these media companies often display brands for advertisement goals. However, the opposite case may also occur in which captured media may contain visual information the owner does not want to display. Therefore we propose a SIFT-based learning algorithm to classify specific logos in a system for an automatic blurring process where a user can select and modify the detected regions. The algorithm uses k-nearest neighbours classification, which is trained on a database of gathered logos, in combination with logo templates. The system is regarded as intuitive to use, the system detects 73% of all cases and predicts the brand correctly 98% of the time.**

Keywords*—Logo recognition, SIFT-feature extraction, k-nearest neighbour classification, template matching, image processing.

I. Introduction

Video material (like movies or documentaries) often contain logos and other products, often placed there explicitly for commercial purposes. This practice goes back to the times of the first movies of Lumière in the 1890s [14]. In the past 30 years there has been paid more attention on the effects of product placement on audiences in scientific literature. There has been research in the recognition and recall of brands by E.M. Steortz, L. A. Babin *et al.*, S. Hong *et al.* [1][8][18] for 'traditional' media such as film but also more recent work such as the effects of product placement in social media by S. H. Liu *et al.* [12]. These studies suggest that viewers do recall and recognize brands and in the case of social media influences the Internet-browsing behavior of the viewer. The research of C. Janiszewski and S. Shapiro *et al.* indicates that even when not recalled explicitly, consumers may still be influenced by exposure to the placement of products or brands [10][11][17].

However, whether intentionally placed or not, there may be several reasons a person might want to blur brands or logos: As a user you might want the decrease the influence of these logos on you, as a producer you might not want to advertise certain companies or as a broadcasting network you might want to blur logos for legal reasons.

In this paper we will take on the problem of logo recognition within images. Our goal is to create an automated blurring system for use on the Internet and analyse its strong and weak points. To recognize logos in images we will use a database of logo images. We would however also like the system to learn new logos that are not yet in the database. We will call these logos *unknown logos*. When the system encounters an unknown logo, we would like the user to verify the brand it represents. Then our system should learn to recognize other instances of this logo and add them to its database. This way it should be able to incrementally increase its scope. Learning logos based on user input and interaction is a novel approach not seen in previous literature at the moment of writing.

The paper is structured as follows; in section II we will give an overview of related works. In section III various aspects of our system are detailed. This is then followed by our methodology of testing in section IV and results in section V. We end the paper with a conclusion and discussion in sections VI and VII respectively.

The authors are Masters degree students in Computing Science (CS) and Game and Media Technology (GMT) at Utrecht University, the Netherlands.
Gary Fibiger (Student no.: 5748143) {g.fibiger@students.uu.nl}
Jinfeng Guo (Student no.: 5610443) {j.guo3@students.uu.nl}
Geerten Klarenbeek (Student no.: 5532442) {g.j.klarenbeek@students.uu.nl}
Joost Linthorst (Student no.: 3282538) {j.j.p.linthorst@students.uu.nl}
Erwin Westra (Student no.: 3492214) {e.a.j.westra@students.uu.nl}

II. Related Work

Automated methods for logo recognition have been studied for some time. Over the years there have been many methods for logo recognition for various types of media. For example, a lot of work has been done on the recognition of logos in paper documents as to more easily identify the source of a document. D.S. Doermann *et al.* used geometric invariants to capture profiles of points of interest in logos [2]. These geometric invariants are not sensitive to scaling, rotating or other geometric transformations. Later they included the use of a hybrid of algebraic and differential invariants [3]. Algebraic invariants are used to represent shapes as simple low order polynomials, but are limited in their expressiveness. Differential invariants are very dependent on local information.

E. Fransesconi *et al.* implemented a logo recognition system using a recursive neural network approach [6]. Here, logo images are structurally represented by contour trees, capturing a topological partitioning on which to compare. The use of a recursive neural network gives a robust recognition method, but an increase in the amount of nodes in a tree structure to capture a profile of a logo makes the learning process harder. It also does not deal well with logos that are topologically similar.

D.G. Lowe introduced an object recognition system for images using the Scale Invariant Feature Transform (SIFT) method [13]. Scale invariant features are found using a staged filtering approach. These features are then grouped together using a nearest neighbor algorithm to find potential object models. When at least three features agree to some kind with such a model then there is a strong indication that that object is in the image. This allows for robust recognition even under partial occlusion. SIFT does not deal well with color or texture differences. A logo with inverted colors would not agree to the model of that same logo with normal colors.

Romberg *et al.* introduced a novel way of indexing image features to effectively find logos in an image. Indexing triangles of feature points that are scale invariant [15]. These triangles contain define the spatial relation between feature points. A class for the logo in classification is then simply a collection of triangles. The precision is dependent on the amount of the constructed triangles per logo and the quality of capturing the actual logo structure.

F. Yang *et al.* present a multiple feature fusion method for logo retrieval [21]. A TF-IDF approach is used for constructing feature vectors using differences between image pairs. A logistic regression model is used for weight learning. N. Farajzadeh presents an exemplar-based method for logo recognition [5]. All logo images, which are preprocessed, are applied rotation and affine to create multiple exemplar images for each logo to simulate the natural image distortions. For feature extraction, Zernike Moments are used for representing highly accurate shapes with little redundancy, and Local Ternary Patterns are used to represent textures. A Support Vector Machine-classifier is used for every exemplar, and a set of simple exemplar SVM detectors is generated. Results show that this method has better recall but is less precise than other methods such as the SIFT-based ones. As a trade-off the method does not need a training set to induce a robust model. F. Wang *et al.* present a method for logo detection in social media data [20]. Dense Kernel Local Binary Pattern features (DKLBP) are extracted to represent objects during both the training session and the detection session. A SVM classifier is used for DKLBP feature classification. They use meta-data associated with an image in social media to offer more precise results without a big reduction in recall. S. Yu *et al.* uses a Bag-of-Words based method to do vehicle logo recognition [22]. In the method, Dense-SIFT features are extracted, and data is trained using a SVM classifier. Features extracted by Dense-SIFT are robust as they are invariant to scaling or rotation. Results show a higher accuracy and less computation time than other methods. A comparison on various moment-based methods for logo recognition is compiled by Z. Zhang *et al.* which classifies all these methods according to applications, which include logos with noise, rotation, scaling and other transformations [23]. Altogether there are various methods devised to try and solve visual recognition problems with SIFT features being a popular building block since its introduction in 1999.

III. SYSTEM OVERVIEW

Ideally, when an image is uploaded to the system the image is checked by our logo recognition algorithm to look for known logos from our database. This database contains all the logo feature profiles that characterizes each individual logo. When the algorithm is finished we present the user the uploaded image with area indications of found logos. The user can then edit this information by discarding indications or adding new areas which contain logos that the algorithm may have missed, as is the case when a logo is not yet included in the database. The user can provide a company name for any

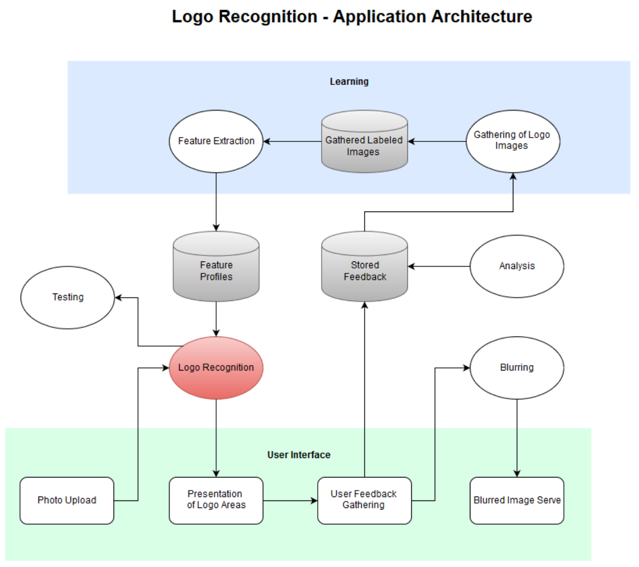


Fig. 1: A schematic overview of the system databases and processes.

user indicated logo. This feedback is used to retrieve training images based on the company name so that the system may learn how to recognize the company. When the user accepts the given results from our algorithm and possibly added their own input the user can request to remove the indicated logo regions. The processed image is then returned to the user without the indicated logos. A schematic overview can be found in Figure 1.

We partially implemented two different algorithms as a proof of concept. The original method in (III-A) proved difficult to implement and as a result performed poorly. The original method has an easily extendable database (although we did not implement this in the online version due to memory constraints on the available server). We also had to reduce the number of logo classes on the online version, again due to memory constraints. Running the classifier offline allows for the full range of classes to be processed. Due to the less than desirable performance of the original method we decided to implement a second, simpler method which we now call the revised method. This method uses a substantial part of the same ideas as the original method but uses a different method of classification. The revised method uses significantly less memory and we did not have to use a reduced logo class set. We also took the opportunity to improve the logo removal process when developing the revised method.

We will now go into further detail about the original method in section III-A, followed by the revised method in III-C, and user feedback processing (III-D).

A. Original method

1) Feature Extraction: For feature extraction a SIFT-based approach is used. The features are extracted using the default method for SIFT feature extraction in OpenCV. SIFT features are image features that do not change when the size (and to a lesser extend the rotation) of the features changes. These features consist of key-points and their accompanying descriptors. The keypoint contains the position, angle and response among other properties. The descriptor is the actual 128-dimensional SIFT descriptor. This descriptor is created by taking a 16x16 neighbourhood of the keypoint and dividing it into 16 4x4 pixel blocks. For each of these pixel blocks an 8-bin histogram is created. The 16 8-bin histograms form the SIFT-descriptor. A visual word is assigned to every keypoint. This visual word is obtained by using the cookbook to find the closest cluster-center (III-A2). To improve the performance of the learning algorithm the features of every image are extracted once and stored in a file for future reference. The database also contains binary masks that indicate where the logo is in the image. Feature files were generated for both full images and masked images.

2) The Cook-book: The cookbook is a dictonary which is used to convert a SIFT-descriptor to a single value. This value is called a Visual Word Label. The cookbook is generated by loading the features from all the images (~3.5 million features) and processing them using a k-means clustering algorithm. The used method is provided by the sklearn package which contains a fast k-mean clustering method for a large amount of samples based on processing mini batches[16]. Generating the cookbook is time consuming (around half an hour), but this only needs to be done once. This step requires over 20GB of memory.

The cookbook does a good job of converting the descriptor to a Visual Word Label if there is a close match to the descriptor. However, the total number of possible descriptors is very large and the number of visual word labels is limited to 2048 which means that the feature space is sparsely populated. This means that a lot of features can get assigned with a label to which it has a large distance. This also means that two descriptors that are actually very similar get a different visual word which are very distant. These different assignments can make it impossible to classify the logo.

3) Triangle detection: descriptor matching: The first step of triangle generation is comparing every pair of two images in a brand. The features in both images are matched with each other using a brute force knn matcher. For every descriptor the best two matches are found in the second image. The matches are filtered based on Lowe's ratio test[13] (i.e. the ratio between the distances of the two best matches must have a ratio smaller than 0.75). This will give us a list of corresponding point pairs.

The amount of mismatches in the filtered set appear to be small. In the event that a mismatch occurs, it is unlikely that those point pairs would be involved in a valid edge (see III-A4) because the angle would be off.

4) Triangle generation: edge creation: The second step of the algorithm is creating edges from the points. An edge is created by taking two points in both images and drawing an edge between them (i.e. both images contain one edge). We create an edge for each point pair which will create $O(n^2)$ edges where n is the number of features. To filter the edges we need to define a similarity measure which is used to accept or reject the edges. The angles α and β (see figure 4) are calculated. These angles are used because they do not change when the logo is rotated. The two edges are compared by α and β by using a similarity method[15]:

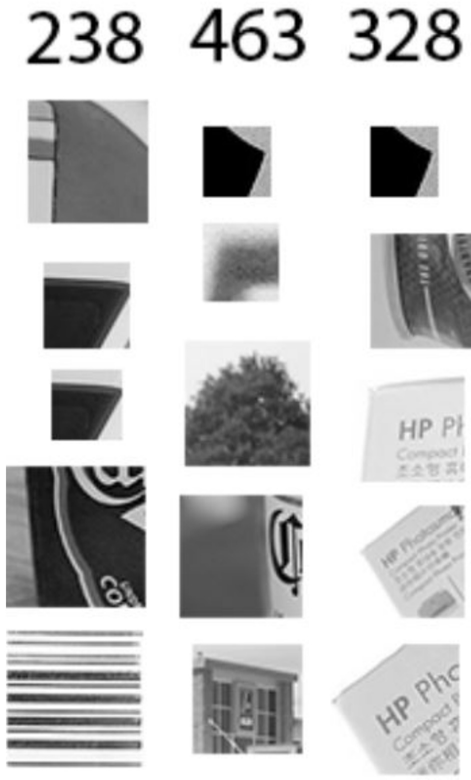


Fig. 2: Multiple features classified with the same Visual Word Labels. The only difference between the first features in 463 and 328 is a difference in the angle of the descriptor.

$$simAngle(\Delta\alpha) = e^{-\Delta\alpha^2/128} \quad (1)$$

The total similarity is calculated by taking the similarity of α and β and multiplying them:

$$simEdge(\Delta\alpha, \Delta\beta) = simAngle(\Delta\alpha) * simAngle(\Delta\beta) \quad (2)$$

If the similarity is smaller than 0.8 the edge is rejected. This will generate a dictionary of edges which point to their counterparts in the second image. At the same time a list of all points pairs that are contained in these edges is maintained.

5) Triangle generation: edge combination: The final step of the triangle generation phase starts with the dictionary of edges and the list of points that were generated in the edge creation step. The algorithm iterates over all the edge pairs and tries to find a point in the point pair list which can be added to the edge to make a triangle. The edge and the point can only create a triangle if the two edges that connect the point to the end points of the edge are also in the edge list. Once the triangle is created, it is checked against a set of requirements to make sure that it is informative:

- The points should be further than 5 pixels apart.

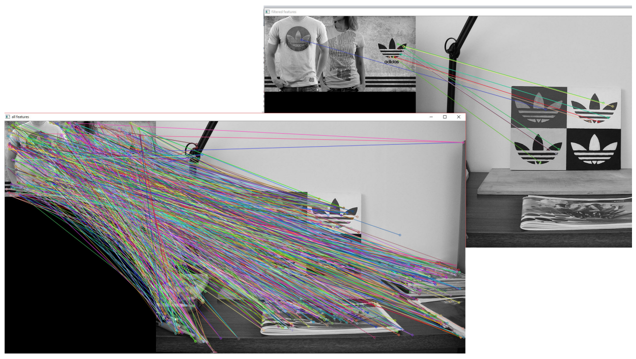


Fig. 3: An example of all descriptor matches versus filtered descriptor matches. Because there are multiple logos in both images there wont be many triangles (two points connected by an edge could be in two different logo's which will make them non-similar), but all descriptor matches are correct.

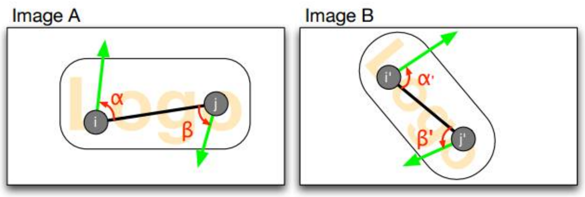


Fig. 4: A schematic overview of an edge.

- The angles of the triangle should be larger than 15 degrees.
- The ratio between the lengths of the sides of the triangle should be in the range [1/3, 3].

The name of the brand is added to the triangles after which they are stored in a file that is used by the learning algorithm.

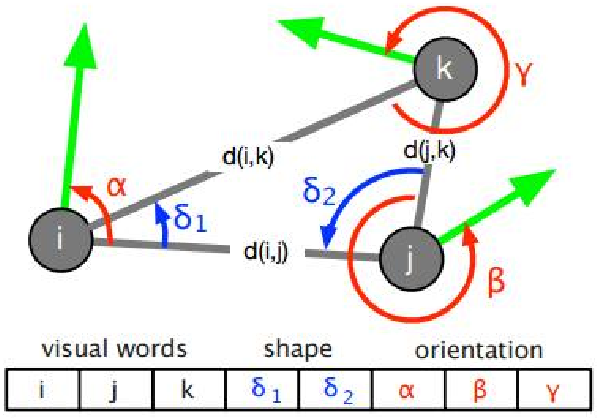


Fig. 5: A schematic overview of a triangle.

6) Learning Algorithm: The database is generated by quantizing the triangles and storing them in a dictionary. The triangles form the keys and point to another dictionary which contains pairs of brands and the number of times that the triangle was found in that brand.

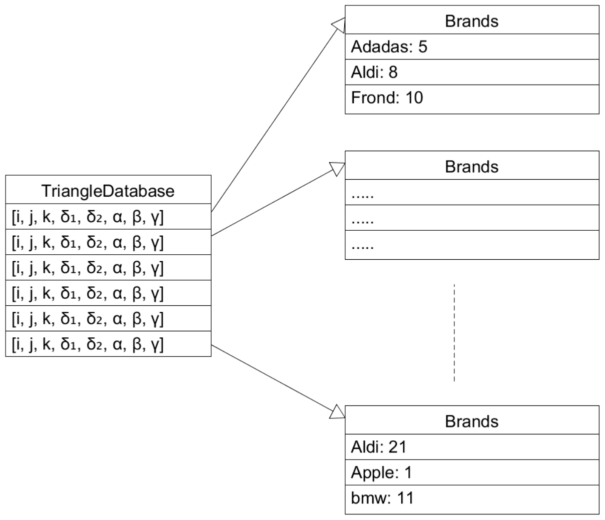

Brands
Adadas: 5
Aldi: 8
Frond: 10
Brands
.....
.....
.....
TriangleDatabase
[i, j, k, δ1, δ2, α, β, γ]
[i, j, k, δ1, δ2, α, β, γ]
[i, j, k, δ1, δ2, α, β, γ]
[i, j, k, δ1, δ2, α, β, γ]
[i, j, k, δ1, δ2, α, β, γ]
[i, j, k, δ1, δ2, α, β, γ]
Brands
Aldi: 21
Apple: 1
bmw: 11

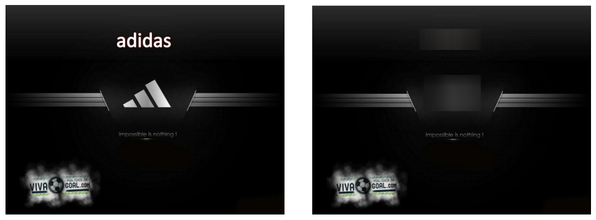
Fig. 6: A schematic overview of the triangle database.

The triangles consist of five angles (see Fig. 5): two shape angles and three orientation angles. The shape angles are quantized into bins of 10 degrees and the orientation triangles into bins of 20 degrees. The angles are binned to their best and second best bin. This results in $2^5 = 32$ combinations that are added to the database per triangle. It would be interesting to see the results if the triangles are quantized to the second best visual word labels too. However, this would increase the memory requirements by another 8 times for the classifier and increase the processing time for the database generation as well.

7) Classification Algorithm: To classify an image we must first extract the SIFT features. The descriptors of these features are then converted to Visual Word Labels. The number of points is limited to the 500 most responsive features to reduce the computation time. The points are sorted by visual word label to ensure the correct order of points within the triangles. All 500 points are combined with a random sample of 200 points taken from the remaining points (with smaller visual word labels). These point pairs are checked against the dictionary of edges. If the point pair is contained in an edge we try to find a third point in a random sample taken from the remaining (smaller) points. These three points are turned into a triangle, quantized, and checked against the triangle database. If the triangle is present we add it to a list of found triangles and accumulate the associated brand votes. The name of the logo is determined by taking the brand with the maximum number of votes divided by their total number of triangles in the database. The triangles are passed on to the region blurring stage.

B. *Region Blurring*

The project requires an image with specific regions blurred, here these would be the potential logo regions. At first, we imagined a nice mask blur function which blurs the logo region and uses a gradient to blend back into the image. This is costly however, so it was decided to constrict the area to the smallest enclosing rectangle. With this, the blurring section downgrades to a much simpler process. Since OpenCV does not have region blur built in, we have to implement this ourselves. OpenCV provides several blur functions, namely the regular blur and the Gaussian blur. Gaussian blur, or Gaussian smoothing, is used widely in image processing for its nice results in eliminating noise and details. We agreed on a way of blurring by first blurring the whole image, and then taking regions from the blurred image and merge it back to the place where it belongs in the original image. This will greatly reduce the diffraction pattern that we get by only blurring the regions. An example is shown in Figure 7.



(a) The original image (b) The processed image.

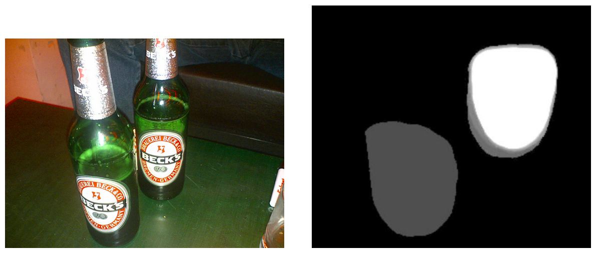
Fig. 7: Image before and after blurring, here only Adidas is considered for blurring.

C. *Revised method*

The second method we developed is basically a reduction of the original method. The method compares unknown images to template images to find logos. The process of classification is similar to the the learning phase of the original method.

1) Learning phase: The learning phase consists of distilling information from the template images. These template images are hand picked to represent the logo well. This means that there should be a variety of perspective in the template images. Some logos can occur inverted (e.g. the Adidas logo appears with a light color on a dark background and vice versa), in this case both of these version should be presented in the templates. The template images are accompanied by image masks. The learning phase finds the keypoints and SIFT-descriptors in the masked image and stores the combination of mask, keypoints and descriptors as a single template. The result of the learning phase is a dictionary of templates stored by brand.

2) Classification: The classification phase starts by reading the template database and the unknown image. Then the SIFT descriptors and keypoints are found in the unknown image. These keypoints are matched against the keypoints in every template and the edges and triangles are created in the same way that this is done in the learning phase of the original method (III-A3). Two time saving measure were introduced by limiting the number of edges to 1000 and skipping the remaining templates for a brand once the number of triangles for that brand exceeds 250. For the system test we classify the image as the brand for which the most triangles are found. To determine the complete area of the logos, every triangle is used as a reference point. Because we know the location of the three points of the triangle on the unknown image and the location of the three corresponding points on the mask we can use an affine transform on the mask to line it up with the unknown logo (these functions are provided by opencv with the functions getAffineTransform and warpAffine). The transformed masks (in the case that multiple triangles are found) are summed together (where a mask pixel is 1 and a non-mask pixel is 0). This will create a heat-map like mask which indicates where the logos are. We use multiple thresholds between the mask being a logo or not a logo. These thresholds are determined per connected region (i.e. each blob is most likely a separate logo) in the mask and is set to the median of the nonzero and non maximum value pixels in each blob.



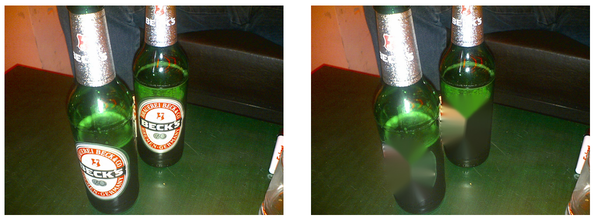
(a) The original image (b) The resulting mask.

Fig. 8: The original image, two logos were found in the image.

3) Logo removal: Because the revised method is able to create a mask for the logos, we can now use a more precise method to remove them. This method of removing uses the pixels around the mask to repair the masked area. The used method is provided by openCV and is called inpaint(). Inpaint uses Telea's method [19] to repair the image. Logos are removed quite unobtrusively if the area around the blur region is uniform in color and has a smooth texture. If this is not the case then unexpected visual artifacts which may be viewed as disruptive. There is no reason that the inpaint function couldn't be used with the original method. The only drawback is that the original method can not provide us with the point triplets of the original triangle locations. Therefore, the mask would have to be a less precise region (e.g. a smallest bounding box).

D. *User Feedback Processing*

For the original method, we can use user feedback to teach the algorithm new logos. Within the system the user can identify areas that contain a logo. The user will be prompted to fill in a company name for that logo so that we may do an image search. For this, we use an automated image



(a) The original image (b) The processed image.

Fig. 9: Image before and after blurring.

retriever with certain search parameters that downloads the first amount of search results from Google Images. The search parameters given to Google are set in such a way that the images gathered this way are of good enough initial quality and generally contain the logo under different orientations and circumstances. These images are then used by our system to try to learn to recognize the corresponding logo.
Using an image gatherer or crawler as was intended in the original method is not suitable in the revised method as is. There is no guarantee or measure on variety and representation of the downloaded images to correctly distill template images. It would still be possible to make the database extendable, but only in an environment with more involved users (e.g. a corporation). In this case the images would be hand picked by the user. The problem with providing an online extendable database in this manner is that the worst case time complexity of the classifier is linear in the number of templates. A malicious user (or a bored teenager) could add an arbitrary number of templates to slow down the classification or add bad templates to decrease is accuracy. Of course there are methods to prevent this, but that is outside of the scope of this project.

IV. Testing

We wanted to measure the following aspects of our system: Logo recognition performance, blurring quality and ease of use of the system. We did both a system perspective and a user perspective test to test these aspects. Note that we have two system methods to test. We did the system perspective test on both system methods to measure the logo recognition performance. We did the user perspective test only on the revised method to measure perceived performance, blurring quality and ease of use. We did not do the user perspective test on the original method because the system perspective test showed that the revised method performed much better.

A. *System Perspective Test*

The system perspective test was done on both system methods. In this test the system tries to find and classify the logos. This test is done without user interaction, which allows us to test many pictures. In the original method we limit the number of features to 500 to make the test faster. This was not necessary for the revised method. We used our database of logo images, which consists of 25 different brands with 70 different photo images for each brand (the images are labeled with the brand). This brings the total number of images to 1750. All photos are taken from a natural environment (so no artificially constructed images). In the original method we used 50 images of each brand for training. In the revised method we manually selected 3 to 6 images to train the classifier. 20 images of each brand were used for testing. For each test image the system tried to find the logo and predict its brand. It sometimes found no triangles and therefore could not detect any logo. We also tested 200 images that contained no logos in both systems. The results of the tests on both system methods can be found in figure 10 and figure 11.

B. *User Perspective Test*

We performed the user perspective test only on the revised method because of its dominating performance in the system perspective test. The user perspective test measures perceived performance, blurring quality and ease of use. We conducted the experiment with 20 test subjects. Each subject was asked to manually download 3 pictures containing logos that our system could recognize. Then they had to blur those logos by going through our system interface. The test subjects evaluated the system by filling out a questionnaire. The questionnaire is based on a 5-point Likert scale. The results can be found in table I.

V. Results

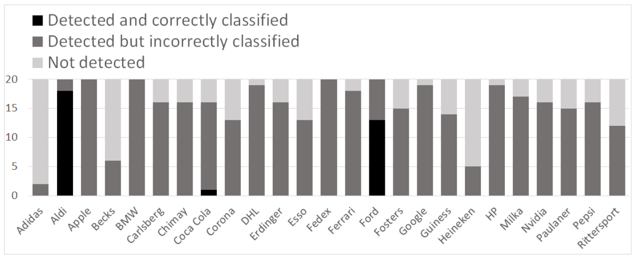


Fig. 10: Results of the original method tested on 20 images of each brand.

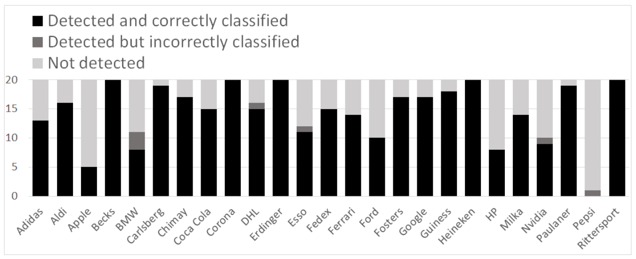


Fig. 11: Results of the revised method tested on 20 images of each brand.

A. System Perspective Test

The results of the system perspective tests are shown in figure 10 and figure 11. Omitted from both figures is that neither method detected any logo in the 200 non-logo images (which is good!). To clearly distinguish between detection and classification we will define recall, precision and accuracy for this paper.

Recall and precision are only concerned with logo *detection* but not with logo *classification*. If the system correctly detects a logo, then it does not matter if it is incorrectly classified. As mentioned earlier no logos were found in the non-logo images. This means that both system methods have a precision of 100%.

Accuracy on the other hand is only concerned with *classification* but not with *detection*. Accuracy ignores the 'Not detected' results. We define the accuracy to be the proportion of correctly classified images of all images where a logo was detected. Note that our precision of 100% is very fortunate with this statistic since we only classify if there is actually a logo present.

Original method: In the result of the original method we can clearly see that only Aldi and Ford are classified well. Adidas, Becks and Heineken were detected but incorrectly classified. With the other brands almost no logos were detected. We noticed during this test that most incorrect classifications actually predicted Aldi. This is probably because Aldi has a large number of triangles (about 10 times the mean number) and therefore a coincidental match with Aldi is likely. This also explains why Aldi is very well classified. The original method has a recall of 30% and an accuracy of 21%. In only 6.4% of all logo-images was the logo detected and classified correctly.

Revised method: The system perspective test for our second version shows much better results. Classification is generally very successful. Detection is poor with Pepsi and Apple. Pepsi was very hard because their logo appears in a lot of different ways. This makes it hard to select a small set of images to cover all occurrences. Apple is hard because it has only a small number of triangles. Other brands were detected quite well. The revised method has a recall of 73% and an accuracy of 98%. In 72% of all logo images was the logo detected and classified correctly. This shows that the revised method performs much better. The large difference between the two methods suggests that the general method is very sensitive to its training data.

B. User Perspective Test

The questions and results of the questionnaire are shown in table I. Sometimes the respondents got the option to answer 'No Opinion' in which case the response is not used. Therefore the number of responses does not have to add up to 20 for each question.

We would like to determine the 'average' answer for each question to determine how people perceive the performance of different aspects of our system. A popular and useful measurement of performance is the confidence interval of the mean, because it takes certainty into account. Note that it is not possible for us to use the mean because the answer options only follow an ordinal scale, not an interval scale. We therefore chose to use a 90% confidence interval of the *median* instead. This gives us the possible categories of the 'real' median with 90% certainty. We can use the binomial distribution to calculate these categories since each response has a 50% chance of landing on either side of the median. The last column of table I displays the categories. Each confidence interval has overlap with two categories in all but one question.

The first block of questions is about the user interface. We can clearly see that the user interface got very positive responses. All questions relating to it had their median confidence interval within the Very Easy and Easy categories.

The next block of questions was meant to determine perceived performance. We can can see that adding blur-areas happened fairly often. Adding a blur area suggests that the system missed a logo in the image. This therefore suggests that the perceived recall was quite low. This is contrast to the 73% recall from our system perspective test. A possible explanation could be that users tend to test images in which the logos are harder to detect. The next question in this block shows that test subjects very rarely had to remove a blur area. Removing a blur area suggests that the system incorrectly detected a logo. The low occurrence rate of this action suggests that perceived precision is high, which corresponds with the high precision from our system perspective test of 100%. The last action, adjusting blur-area borders, occurred fairly rarely. This suggests that a logo is fairly often completely detected instead of only partly. This is a very important finding because we did not test with our system perspective test if the entire logo was found.

The last two question blocks show that testers were fairly positive about the blurring quality and the system in general.

VI. Conclusion

We presented a system to automatically detect and recognize logos in images. We tested two methods of logo recognition against each other. We determined that the method which uses fewer, manually selected training images works far better than the method which simply uses a lot of training images. This suggests that this system is sensitive to its training data. Our revised method finds logos with very high precision, overcoming some of the problems of the original method where the recall and precision suffered greatly from wrong classification due to confusing perceived commonalities. Although recall improved greatly, the system did still overlook logos sometimes meaning the algorithm still has room for improvement.

We were successful in building an interface that is easy to use. Our implemented blurring was considered fine. In general our user were fairly positive about using our system for blurring logos.

VII. Discussion

The SIFT implementation was based on the work of Romberg *et al.* [15] and in hindsight this paper was not specific

How did you experience performing the following operations?	Very easy	Easy	Neutral	Difficult	Very difficult	Median 90% c.i.
Navigate through the steps	13	7	0	0	0	Very easy, Easy
Upload an Image	13	5	1	1	0	Very easy, Easy
Add blur-areas	7	9	2	0	0	Very easy, Easy
Remove blur-areas	6	3	1	0	1	Very easy, Easy
Adjust blur-area borders	10	7	1	1	0	Very easy, Easy

How often did you have to perform the following operations?	Never / Very rarely	Rarely	Neutral	Often	Always / Very often	
Add blur-areas	2	3	6	6	3	Neutral, Often
Remove blur-areas	16	2	0	2	0	Never / Very Rarely
Adjust blur-area borders	3	9	3	2	3	Rarely, Neutral

	Very bad	Bad	Neutral	Good	Very good	
Did the blurred area fit nicely into the picture ?	0	1	6	11	2	Neutral, Good

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	
I would use this software for blurring logos	1	3	8	6	2	Neutral, Agree

TABLE I: Results of the questionnaire on 20 people.

enough on certain details. For instance if certain image preprocessing is preferred or if there is a focus or filtering on certain image features. We managed to build a good system in the end, but this is was achieved via our own experimentation. The original method shows that our initial system based on this paper was not successful. the revised method might be further improved by using parts of the implementation of Romberg *et al.* [15] that were not mentioned in their paper.

We also leave a lot of room for additional features open in our method. We save all user input which contains valuable information which we believe could be exploited. Users constantly upload pictures with logos to our system. They even manually correct all mistakes made by our system by removing and adding blur-areas. This gives direct feedback to our system about which logos it missed. This might be used to improve detection of known logos or even learn new logos. This would be very interesting for future work.

In the beginning of the project Python was chosen for its flexibility in programming, rapid prototyping and relatively known amongst members of the team. For a future revision writing the system in a compiled language like C++ will most likely greatly benefit general computation speed. Another part that could speed up the classification process is making it concurrent. As all logo classes need to be checked to give the most accurate results each class could be checked on a separate thread or cluster in a distributed set-up.

It might also be interesting to look into an alternative for the SIFT method, a promising alternative could be SURF [4][7][9], which has various applications but does not seem to have been used yet for logo identification. We feared that SURF would give too many features that we would not need and would eventually slow down the learning and classification. But with good heuristics in place for what kind of features we extract the overall speed advantage of SURF over SIFT could be very beneficial to the computation time if implemented in our system.

Acknowledgment

The authors would like to thank Christian Eggert of the university of Augsburg in Germany for allowing the use of the logo image database.

References

[1] Laurie A Babin and Sheri Thompson Carder. “Viewers’ recognition of brands placed within a film”. In: *International journal of advertising* 15.2 (1996), pp. 140–151.

[2] David S Doermann, Ehud Rivlin, and Isaac Weiss. “Logo recognition using geometric invariants”. In: *Document Analysis and Recognition, 1993., Proceedings of the Second International Conference on*. IEEE. 1993, pp. 894–897.

[3] David Doermann, Ehud Rivlin, and Isaac Weiss. “Applying algebraic and differential invariants for logo recognition”. In: *Machine Vision and Applications* 9.2 (1996), pp. 73–86.

[4] Geng Du, Fei Su, and Anni Cai. “Face recognition using SURF features”. In: *Sixth International Symposium on Multispectral Image Processing and Pattern Recognition*. International Society for Optics and Photonics. 2009, pp. 749628–749628.

[5] Nacer Farajzadeh. “Exemplar-based logo and trademark recognition”. In: *Machine Vision and Applications* 26.6 (2015), pp. 791–805.

[6] Enrico Francesconi et al. “Logo recognition by recursive neural networks”. In: *Graphics Recognition Algorithms and Systems*. Springer, 1998, pp. 104–117.

[7] Wei He et al. “Surf tracking”. In: *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE. 2009, pp. 1586–1592.

[8] Soonkwan Hong, Yong Jian Wang, and Gilberto De Los Santos. “The effective product placement: Finding appropriate methods and contexts for higher brand salience”. In: *Journal of Promotion Management* 14.1-2 (2008), pp. 103–120.

[9] Zhang Huijuan and Hu Qiong. “Fast image matching based-on improved SURF algorithm”. In: *Electronics, Communications and Control (ICECC), 2011 International Conference on*. IEEE. 2011, pp. 1460–1463.

[10] Chris Janiszewski. “Preattentive mere exposure effects”. In: *Journal of Consumer Research* (1993), pp. 376–392.

[11] Chris Janiszewski. “Preconscious processing effects: The independence of attitude formation and conscious thought”. In: *Journal of consumer research* (1988), pp. 199–209.

[12] Su-Houn Liu, Chen-Huei Chou, and Hsiu-Li Liao. “An exploratory study of product placement in social media”. In: *Internet Research* 25.2 (2015), pp. 300–316.

[13] David G Lowe. “Object recognition from local scale-invariant features”. In: *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*. Vol. 2. Ieee. 1999, pp. 1150–1157.

[14] Jay Newell, Charles T Salmon, and Susan Chang. “The hidden history of product placement”. In: *Journal of Broadcasting & Electronic Media* 50.4 (2006), pp. 575–594.

[15] Stefan Romberg et al. “Scalable logo recognition in real-world images”. In: *Proceedings of the 1st ACM International Conference on Multimedia Retrieval*. ACM. 2011, p. 25.

[16] David Sculley. “Web-scale k-means clustering”. In: *Proceedings of the 19th international conference on World wide web*. ACM. 2010, pp. 1177–1178.

[17] Stewart Shapiro, Deborah J MacInnis, and Susan E Heckler. “The effects of incidental ad exposure on the formation of consideration sets”. In: *Journal of consumer research* 24.1 (1997), pp. 94–104.

[18] Eva Marie Steortz. *The cost efficiency and communication effects associated with brand name exposure within motion pictures*. West Virginia University, 1987.

[19] Alexandru Telea. “An image inpainting technique based on the fast marching method”. In: *Journal of graphics tools* 9.1 (2004), pp. 23–34.

[20] Fanglin Wang et al. “Logo information recognition in large-scale social media data”. In: *Multimedia Systems* (2014), pp. 1–11.

[21] Fan Yang and Mayank Bansal. “Feature Fusion by Similarity Regression for Logo Retrieval”. In: *Applications of Computer Vision (WACV), 2015 IEEE Winter Conference on*. IEEE. 2015, pp. 959–959.

[22] Shuyuan Yu et al. “Vehicle logo recognition based on bag-of-words”. In: *Advanced Video and Signal Based Surveillance (AVSS), 2013 10th IEEE International Conference on*. IEEE. 2013, pp. 353–358.

[23] Zili Zhang et al. “A Comparison of Moments-Based Logo Recognition Methods”. In: *Abstract and Applied Analysis*. Vol. 2014. Hindawi Publishing Corporation. 2014.