Math. Struct. in Comp. Science (2014), vol. 24, iss. 5, e240506, 10 pages. © Cambridge University Press 2014
doi:10.1017/S0960129513000820

Method of item recognition based on SIFT and SURF†

WENYU CHEN, WENZHI XIE and RU ZENG

School of Computer Science and Engineering,
University of Electronic Science and Technology of China,
Sichuan Chengdu 610054, P.R.China
Email: cwy@uestc.edu.cn;xiewenzhi1008@qq.com;zengru_2005@qq.com

Received 20 July 2011; revised 18 August 2013

Item recognition has become a hotspot in the field of computer vision research. SIFT has the advantage of requiring a low amount of information, a fast running speed and high precision, but it requires large data calculations and thus takes a long time to perform the item recognition. In this paper we propose a method of item recognition based on SIFT and SURF that provides a new way to solve the problem of item recognition, and has both feasibility and availability. This technique currently ignores colour information when dealing with colour images, but the evaluation method is capable of taking colour quality characteristics into account so it should be possible to improve the algorithm in the future. Experimental results show that this system of item recognition based on the SURF algorithm gives better matching recognition, is faster and has greater robustness.

1. Introduction

Item recognition based on reference images to identify the marked items in a target image is an important technology in the field of machine vision. In recent years, image-based item recognition technology has become a research hotspot in the field of machine vision and pattern recognition. The methods used for item recognition can be divided into two classes: methods based on image features and methods based on area matching (Brown 1992). The SIFT algorithm (Lowe 1999) belongs in the former class and has higher efficiency at present. As an improvement to the SIFT algorithm, Bay *et al.* (2008) proposed the SURF algorithm, which boosted its performance.

2. Improving the SIFT algorithm

The SIFT algorithm was proposed by D. G. Lowe in 1999. It is a feature description method with good robustness and scale invariance (SIFT stands for scale-invariant feature transform). However, it requires large data calculations and has high time complexity, so the computations take a long time to complete. Over time, several improvements have been proposed. Ke and Sukthankar (2004) used the PCA-SIFT method to reduce the dimension of the feature description, but without any prior knowledge, there was an

† This work was supported by the Chinese National Science Foundation Grant Number 61073120

https://doi.org/10.1017/S0960129513000820 Published online by Cambridge University Press

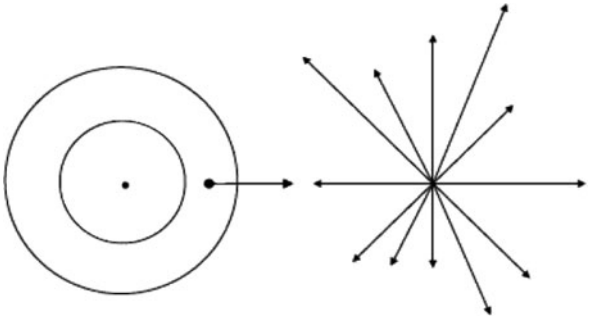


Fig. 1. Feature descriptor for improved SIFT algorithm

increase in the size of the calculations. Delponte *et al.* (2006) used the SVD method to match features, but the process of matching requires complicated calculations and cannot be applied to wide-baseline matching. Grabner *et al.* (2010) used an integral image to improve the speed of SIFT computation, but at the cost of reduced quality. These methods made improvements to the feature description or matching stage, but did not change the algorithm itself.

The feature points for the SIFT algorithm are stable, but the process of generating the feature descriptor using the main rotation direction and counting the gradient histogram produces a 128-dimensional vector. This method requires large calculations and takes half of the algorithm's run time. Moreover, when the image registration is complete, the computing cost of feature matching for the 128-dimensional vector is large. These are the main areas affecting the real-time use of the SIFT algorithm. Therefore, the main aim for improving the algorithm is to reduce the amount of calculation required to produce the feature descriptor and to enhance the speed of feature matching.

Experiments show that when the SIFT matching algorithm is used to match feature points, the step of producing the feature descriptor takes 60%–80% of the time. To improve the algorithm, we combine feature point direction assignment and feature descriptor production into a single step, and replace the 128-dimensional vector with a 64-dimensional vector, while at the same time using a traverse search in the matching process to find the nearest neighbour and next nearest neighbour feature points to the sample feature point.

The improved SIFT algorithm is as follows:

(i) Build the scale space and locate the feature points.

(ii) Create a circular window with the feature point as centre and a radius of 4σ as recommended by Lowe, and divide it into two concentric rings. Then use the 32 gradient directions to produce 32 feature vectors in each ring. Figure 1 shows that the external ring produces feature vectors in 32 directions.

(iii) Compute the gradient model and direction for each pixel in each ring. Then, as in the standard SIFT algorithm, use the gradient histogram to calculate the accumulated value of the inner ring to give the vector $(d_1, d_2, ..., d_{32})$ and the accumulated value of the external ring to give the vector $(d_{33}, d_{14}, ..., d_{64})$ – note that each of the d_i is itself a vector.

https://doi.org/10.1017/S0960129513000820 Published online by Cambridge University Press

(iv) Normalise each of the feature vectors produced by the two rings separately – this can reduce the effect of illumination differences. For example, if D is the feature vector for the inner ring, that is $(d_1, d_2, ..., d_{32})$, after normalisation we get

$$\bar{D} = \frac{D}{\sqrt{\sum_{i=1}^{32} d^2{}_i}} = (\bar{d}_1, \bar{d}_2, ..., \bar{d}_{32}). \quad (1)$$

(v) Find the largest vector in $\bar{d}_i$ for each ring and cyclically permute $\bar{D}$ so that the largest vector appears first to ensure there will be rotational invariance in the matching process. For example, if $\bar{d}_5$ is the largest vector in the inner ring, that is,

$$\bar{d}_5 = \max\{\bar{d}_1, \bar{d}_2, ..., \bar{d}_{32}\},$$

after the cyclic permutation, we get

$$(\bar{d}_5, \bar{d}_6, ..., \bar{d}_{32}, \bar{d}_1, ..., \bar{d}_4).$$

(vi) Use the traverse search algorithm to search for the nearest and next nearest neighbour points to perform the matching.

The improved SIFT matching algorithm has lower complexity and a reduced time for subsequent matches. Also, because we use a ring model rather than a circle, the robustness of the algorithm is enhanced.

3. Building the integral image

The idea and general outline of the SURF algorithm is similar to SIFT, but there are differences of detail throughout the whole process.

The performance of the SURF algorithm can be greatly improved by using an integral image. The integral image can be quickly obtained from the input image and it speeds up the process of computing the sum of values in any rectangular area. Given an input image I, the value at point $X(x, y)$ in the integral image $I_\Sigma(X)$ is the sum of all the pixels in I within the rectangular area formed by the origin of image I and the point $X(x, y)$ with sides parallel to the axes, that is (see Figure 2)

$$I_\Sigma(x, y) = \sum_{i=0}^{i \leqslant x} \sum_{j=0}^{j \leqslant y} I(i, j). \quad (2)$$

As mentioned above, the advantage of using the integral image in the algorithm is that the sum of all the pixel values within any rectangular area of the original image can be calculated by simply adding and subtracting the values at the vertices in the integral image. For example, the sum of the grayscale values in the rectangular area fixed by the four vertices A, B, C, D in Figure 2 is given, for any rectangle in the integral image, by the formula

$$\sum\nolimits_S = A + D - (B + C) \quad (3)$$

where A, B, C and D are the values of the integral image at those points.

https://doi.org/10.1017/S0960129513000820 Published online by Cambridge University Press

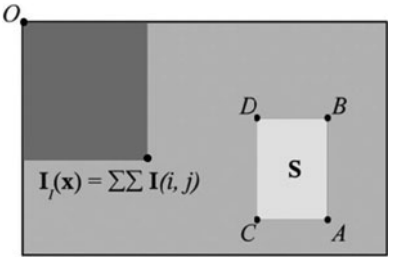


Fig. 2. Schematic diagram for building an integral image

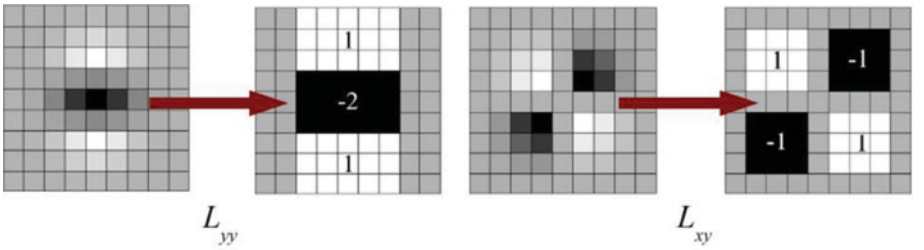


Fig. 3. (Colour online) Schematic of square filter

The use of the integral image in the feature point detection and feature description steps greatly increases the speed of the SURF algorithm.

4. Rapid Hessian detection

Feature point detection in the SURF algorithm is based on the value of the Hessian determinant of a matrix. When the value of the Hessian determinant of a matrix is a local maximum, the detecting point is a position of interest. That is to say, the pixel point of this point is a blob-like image structure, that is, a small area consisting of the feature point and the small area near to the feature point that is lighter or darker than the surrounding area in the image.

Given a pixel point $p = (x, y)$ in an image, the Hessian matrix under the scale of the point $p = (x, y)$ is defined by

$$H(P, \sigma) = \begin{bmatrix} L_{xx}(P, \sigma) & L_{yx}(P, \sigma) \\ L_{xy}(P, \sigma) & L_{yy}(P, \sigma) \end{bmatrix} \tag{4}$$

where $L_{xx}(P, \sigma), L_{xy}(P, \sigma), L_{yy}(P, \sigma)$ are the convolutions of the image with second-order Gaussian partial derivatives at the point p – these are derived using the Laplacian of Gaussian filter.

In Figure 3, the filter template is the 9×9 square filter, which is similar to a Gaussian filter with σ set to 1.2. This represents the minimum scale for calculating the blob response. After using this square filter, the Hessian determinant of the matrix is defined by

$$\det(H_{approx}) = D_{xx}D_{yy} - (\omega D_{xy})^2 \tag{5}$$

https://doi.org/10.1017/S0960129513000820 Published online by Cambridge University Press

where D_{xx}, D_{yy} and D_{xy} represent the approximate second-order partial derivatives of the Gaussian function in the x, y and xy directions, respectively, and ω is a weight coefficient to balance the value of the Hessian determinant of the matrix to ensure energy conservation between the approximate Gaussian kernel function and the true Gaussian kernel function. The approximate value of the weight coefficient ω is given by

$$\omega = \frac{|L_{xy}(1.2)|_F |D_{yy}(9)|_F}{|L_{yy}(1.2)|_F |D_{xy}(9)|_F} = 0.912... \approx 0.9 \tag{6}$$

where $|...|_F$ is the Frobenius norm. Hence, the value of ω depends in theory on the chosen scale size, but this variation is meaningless for practical feature detection, so we just use the constant value 0.9 for ω, so equation (5) becomes

$$\det(H_{approx}) = D_{xx}D_{yy} - (0.9D_{xy})^2. \tag{7}$$

Experimental data in the literature (Ke and Sukthankar 2004) show that using a square filter to approximate a Gaussian filter does not lead to any decrease in the speed and effectiveness of the convolution operation. This approximate value for the Hessian determinant of the matrix represents the blob response at the point $X = (x, y, \sigma)$. The detected feature point is a local maximum of the blob response under different scales.

5. Building the SURF scale space

Two images with the same content should be correctly matched even if they have different sizes. To achieve this, the detection operator must be able to detect feature points having the same physical position in different scale spaces. The scale space is generally constructed using an image pyramid where the Gaussian function is applied with different scale factors and the image is input as a convolution. As shown in Figure 4, the higher levels in the image pyramid can be obtained by down sampling the original image. Lowe's SIFT feature extraction algorithm used DoG (Difference of Gaussian) images by subtraction of adjacent layers in the image pyramid, and fixed points of interest by local maxima or minima in the DoG image.

Unlike the SIFT feature extraction algorithm, SURF uses the original image directly with different sizes of square filter. Because the integral image is used in the process, the calculations for the different sizes of square filter can all be carried out at the same speed.

We begin by dividing the scale space into many layers. Each layer includes a series of response images which are obtained after Gaussian convolution with a square filter with gradually increasing original image and size. Each layer has a scale with a span count of two, and each layer is then divided into a fixed number of scale values. Because the integral image is discrete, the minimum difference between two adjacent scales is determined by the number of positive and negative values in the direction of the second-order partial derivative.

The first layer of the scale space is produced using an approximate Gaussian convolution template to carry out the convolution operation on the original image. Subsequent layers are then produced by convolution operations on the image using templates with gradually increasing sizes. The scale interval of the base layer is set to 6, and then doubled at each

https://doi.org/10.1017/S0960129513000820 Published online by Cambridge University Press

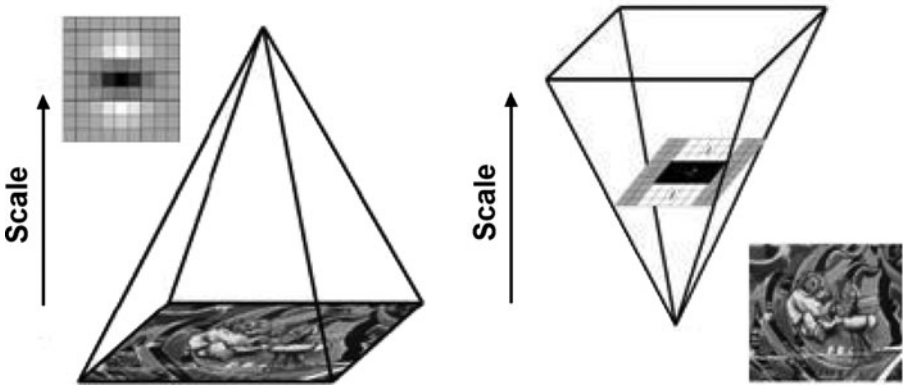


Fig. 4. Image filtering pyramid

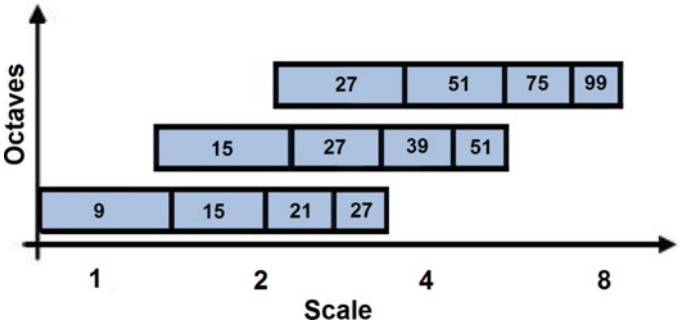


Fig. 5. (Colour online) Size distribution of square filter of former three groups

subsequent layer (12, 24, and so on). This ensures that there is always an integer number of pixels in each template with a single pixel at its centre. The output of the above 9×9 filter is taken as the initial scale layer, which we will refer to as having a scale s of 1.2 (corresponding to Gaussian derivatives with σ equal to 1.2). As shown in Figure 5, the sizes of the square filters on the top layer are 9, 15, 21, 27, the sizes on the second layer are 15, 27, 39, 51, and the sizes on the next layer are 27, 51, 75, 99.

If the size of the square filter is still less than the size of original image, we can produce a fourth group of square filters 51, 99, 147, 195 defined by formula (7). The size of each layer can be obtained by the formula

$$\begin{aligned} \sigma_{\text{approx}} &= \text{Current Filter Size} \times \frac{\text{Base Filter Scale}}{\text{Base Filter Size}} \\ &= \text{Current Filter Size} \times \frac{1.2}{0.9} \end{aligned} \tag{8}$$

To find a feature point on each image layer in the pyramid, the SURF feature detection algorithm uses non-maximum suppression within the field. When the Hessian determinant of the matrix is a maximum, we use the interpolation method proposed by Brown to interpolate the image space and scale. Finally, we can fix the scale and position of the

https://doi.org/10.1017/S0960129513000820 Published online by Cambridge University Press

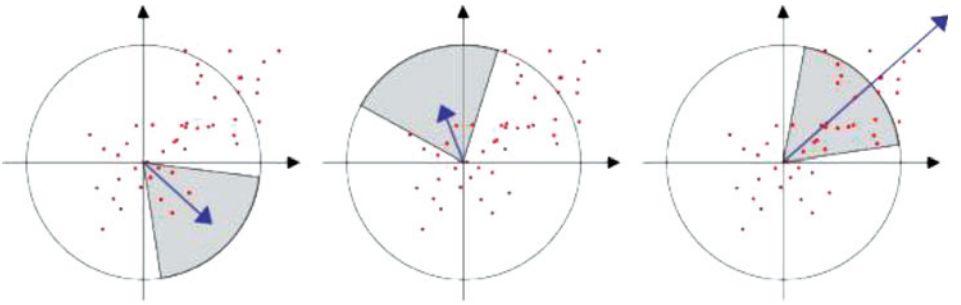


Fig. 6. (Colour online) Schematic showing how the direction of an image feature point is calculated

feature point. Because the scale for the first layer of each image is very large, the scale space interpolation method is very important in the SURF feature detection algorithm.

6. Image feature description

We first use information in the circular region near to the image feature point to get the direction of the image feature point. We then construct a rectangular area in the direction of the image feature point. Finally, we extract the feature descriptor from the constructed area. To get feature descriptors with rotational invariance, we compute the Haar wavelet response of points located in a circular field drawn around the feature point with radius $6s$ (where s is the scale value of the feature points) in the x and y directions. We again use the integral image here to filter quickly. We need to extract the principle direction under all kinds of changes to guarantee the rotational invariance of the descriptor.

After computing the Haar wavelet response, we produce a GIASS circle centred on the feature point as shown in Figure 6, where the dots represent the Haar wavelet response within the GIASS circle. The predominant direction is then calculated using a sliding sector with angle $\pi/3$ (shown in gray in Figure 6) to scan the whole circle. By adding all the Haar wavelet responses in the x and y directions, we can get a temporary direction vector. We then choose the longest temporary direction vector as the predominant direction of this image feature point.

After determining the predominant direction of the image feature points, the feature point descriptors are produced using these directions. Taking the feature point as the origin and the predominant direction to determine the coordinate axis, a square is constructed with sides equal to the scale corresponding to the image feature point. This square area is then subdivided into 4×4 subdomains, and the Haar wavelet response of $5s \times 5s$ sampling points, where s is the value of the sampling step, is computed in each subdomain.

For computational convenience, the Haar wavelet response in the horizontal and vertical directions relative to the main direction are represented by d_x and d_y, respectively, and the matching weight coefficient is also assigned to the response value. We then compute the sum of the d_x, the sum of the d_y and the sums of their absolute values. These are then combined to give the characteristic feature vector for the subdomain:

$$V_{svb} = \left(\sum d_x, \sum |d_x|, \sum d_y, \sum |d_y| \right).$$

https://doi.org/10.1017/S0960129513000820 Published online by Cambridge University Press

The dimension is 4 in each subdomain, so combining the vectors for all 4×4 subdomains gives a characteristic vector with dimension 64. The characteristic vector is then normalised to give the descriptor some invariance against illumination change.

7. Image feature matching

After identifying the features points in the image, the next task is feature point matching. The methods for matching feature points usually involve unchanged distances, related coefficients, Hausdorff distances (Niu *et al.* 2010), or other kinds of similarity and distance measures.

Because the information about the image feature point neighbourhood is included in the feature vector, potential matching pairs of feature points can be found using the vector nearest distance matching method so no extra computation is needed. Here we use the Euclidean distance as the initial matching judgement measure for two images.

In order to identify matching image features, we take feature points in the item identification image and compare their separation, measured by Euclidean distance, with the distance between pairs of points in the matching image. We use 60% as the matching threshold, so if the difference between the distances is less than 60%, we regard it as a match. If we used a lower threshold, we would get fewer, but more reliable matches.

However, the matching method described above is not perfect and may produce incorrect matching pairs. To reduce the number of errors, we combined it with the RANSAC (Random Sample Consensus) method (Fischler and Bolles 2010) to calculate the relationship between the images in terms of a transformation. The basic idea is as follows:

(1) Take n random samples from three groups of corresponding point pairs and compute the M matrix. For each group, compute the corresponding distances d. Compare these using the threshold method, and class those consistent with M as inner points.
(2) Select a point set that includes the largest number of inner points – if there is more than one set with the largest number of inner points, select one with the smallest standard deviation.
(3) Use the selected point set to recalculate matrix M, and minimise the least square error. In calculating the final solution, remove any outer points that do not conform with the relationship satisfied by most points to eliminate their influence on the matching.

8. Analysis of experimental results

We ran simulations on a PC with an AMD 5000+ CPU and 2G of memory running Windows XP, and we used VC++6.0 to develop the software.

To validate the superiority of the SURF algorithm and show that SURF is an effective algorithm for extracting item features, we carried out feature detection for several different items.

The time taken for this algorithm to perform one matching is about 100ms, and the accuracy rate is greater than 80%. We used the algorithm to detect different items – the results are shown in Figure 7.

https://doi.org/10.1017/S0960129513000820 Published online by Cambridge University Press

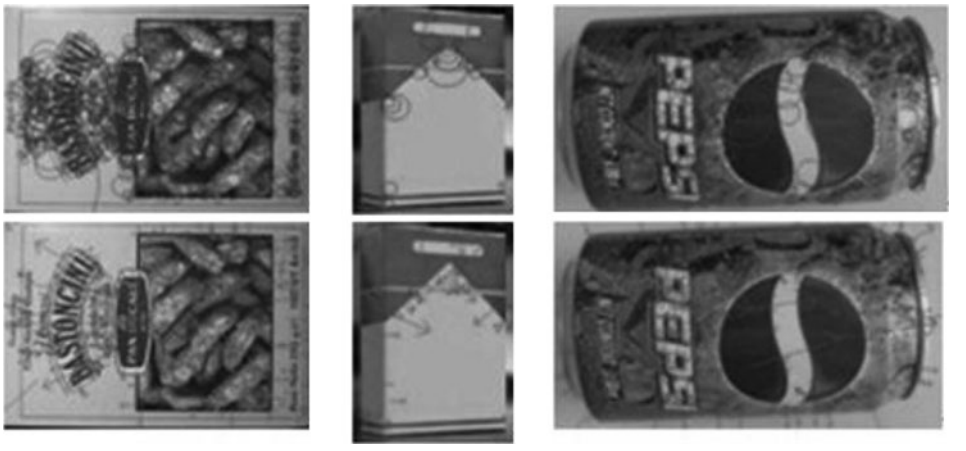


Fig. 7. The effect of feature detection

Table 1. *Performance comparison for SURF and SIFT when recognising different items*

SURF	Number of descriptors	311	59	450
	Detection time /ms	107	54	173
SIFT	Number of descriptors	763	159	1230
	Detection time /ms	453	203	603

The experimental results show that the SIFT algorithm produces a larger number of feature descriptors than the SURF algorithm, but it is much slower and generated a large number of ineffective feature points, which is not a problem the SURF algorithm suffered from. The tests showed that the SURF algorithm is superior to the SIFT algorithm in terms of responsiveness. Comparing the numbers of feature points and detection times in Table 1 when the algorithms are applied to the same image, we can conclude that although SURF produces fewer feature points, it has better matching pair performance using the nearest matching method. The time taken by the SURF algorithm is only one third of the time taken by SIFT, and it is more accurate as well as being more responsive.

Figure 8 shows item matching using the SURF algorithm in different experimental conditions.

9. Conclusions

We have developed a new method of item recognition based on SIFT and SURF that satisfies the demands of accuracy and responsiveness in item recognition. We have elaborated on the basic principles of the SURF algorithm by:

(1) using the SURF algorithm to extract feature points in the item image;
(2) using the Euclidean distance method to find corresponding points of interest in the images;

https://doi.org/10.1017/S0960129513000820 Published online by Cambridge University Press

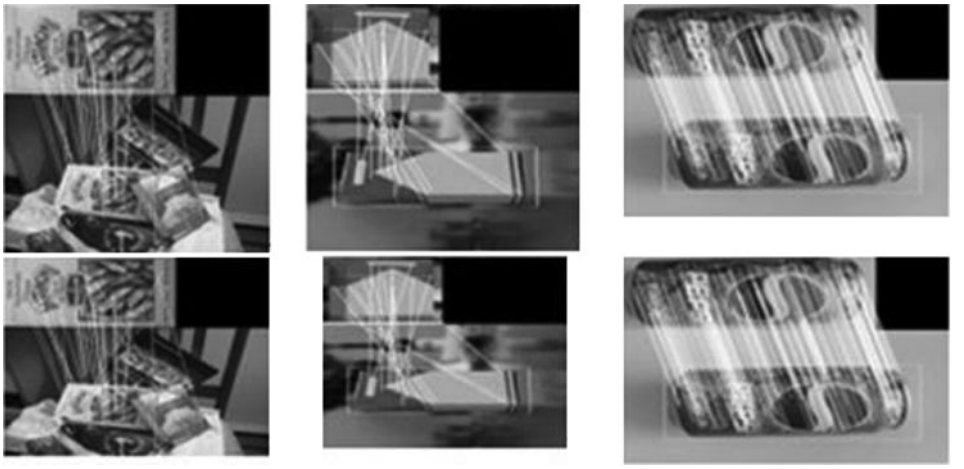


Fig. 8. Recognition results for different items

(3) adding the RANSAC (Random Sample Consensus) method to find the mapping relationship between images, which is then evaluated to identify items in the images.

We have shown that this method is both feasible and practical.

References

Bay, H., Ess, A., Tuytelaars, T. and Gool, L. V. (2008) Speeded-Up Robust Features (SURF) – preprint.

Bay, H., Tuytelaars, T. and Gool, L. V. (2006) SURF: speeded up robust features. *Proceedings of the 9th European Conference on Computer Vision* 404–417.

Brown, J. G. (1992) A survey of image registration techniques. *ACM Computing Surveys* **24** (4) 325–376.

Delponte, E., Isgrò, F., Odone, F. and Verri, A. (2006) SVD-matching using SIFT features. *Graphical Models* **68** (5) 415–431.

Fischler, M. A. and Bolles, R. C. (2010) Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM* **24** (6) 381–395.

Grabner, M., Grabner, H. and Bischof, H. (2010) Fast approximated SIFT. *Proceedings of the Asian Conference on Computer Vision* **1** 918–927.

Ke, Y. and Sukthankar, R. (2004) PCA-SIFT: a more distinctive representation for local image descriptors. *Computer Vision and Pattern Recognition, 2004. Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 511–517.

Lowe, D. G. (1999) Object Recognition from Local Scale-Invariant Features. *Computer Vision 1999: Proceedings of the International Conference on Computer Vision* 1150–1157.

Niu, L.-P., Mao, S.-Y. and Chen, W. (2010) Image registration based on Hausdorff distance. *Journal of Electronics and Information Technology* **36** (11) 2247–2250.

https://doi.org/10.1017/S0960129513000820 Published online by Cambridge University Press