

Chess Piece Recognition Using Oriented Chamfer Matching with a Comparison to CNN

Youye Xie¹, Gongguo Tang¹, William Hoff²

¹Department of Electrical Engineering, Colorado School of Mines, Golden, Colorado USA

²Department of Computer Science, Colorado School of Mines, Golden, Colorado USA

{youyexie, gtang, whoff}@mines.edu

Abstract

Recognizing three dimensional chess pieces using computer vision is needed for an augmented reality chess assistant. This paper proposes an efficient 3D pieces recognition approach based on oriented chamfer matching. During a real game, the pieces might be occluded by other pieces and have varying rotation and scales with respect to the camera. Furthermore, different pieces share lots of similar texture features which makes them more difficult to identify. Our approach addresses the above problems and is capable of identifying the pieces with different scales, rotation and viewing angles. After marking the possible chessboard squares that contain pieces, the oriented chamfer scores are calculated for alternative templates and the recognized pieces are indicated on the input image accordingly. Our approach shows high recognition accuracy and efficiency in experiments and the recognition process can be easily generalized to other pattern recognition applications with 3D templates. Our approach outperforms the convolutional neural networks under severe occlusion and low resolution conditions and has comparative processing time while avoids the time consuming training process.

1. Introduction

Augmented reality (AR) can greatly improve the effectiveness of people in work and play. It can automatically recognize objects using computer vision techniques and display graphical augmentation registered to the object, to provide guidance and instruction. AR has been widely applied in education [1], industrial design and medical treatment [2]. AR can also help people learn the game of chess, a popular intellectual and entertaining game all over the world. For example, the system could display allowable moves as an overlay on an image of the board, using either a hand-held or a head-mounted display. In order to do this, a chess AR system must first recognize the chessboard and

the chess pieces, from a mobile hand-held or head-mounted camera, and locate the pieces on the board. The task can be challenging if the board is viewed from a low viewing angle, instead of directly overhead. This may cause pieces to partially occlude each other. Additionally, some pieces are highly similar to each other, such as the rook and pawn, which may lead to misidentification.

This paper focuses on the problem of recognizing different 3D chess pieces from a single image of the chessboard, under game conditions. We use a chamfer matching approach, which permits flexible operating angles and allows for different occlusion conditions. Furthermore, our method has potential in other applications. For example, in many industrial applications, the objects to be recognized are small with relatively little image texture [3] and CAD models are often not available or are difficult to obtain. In these cases, taking a small number of training images is feasible and our method is applicable to these problem domains. The paper is organized as follows. In section 2, we describe related work. In sections 3 and 4, we present our approach for chessboard and chess piece recognition, respectively. In section 5, we show experimental results and a comparison to an alternative approach using convolutional neural networks (CNNs). We conclude this paper in section 6.

2. Related Work

Many algorithms have been developed to recognize a chessboard for the purpose of camera calibration and 3D scene reconstruction. Most of these use the approach of detecting corners on the board [4, 5]. However, when the chessboard is populated with chess pieces, such as during an actual game, many corners might be occluded by pieces. Therefore, algorithms for recognizing populated chessboards typically use line detection based methods [6, 7, 8].

The research on chess piece recognition is sparse. Early approaches modified the chessboard and pieces with sensors [9]. However, modified chessboards and pieces are

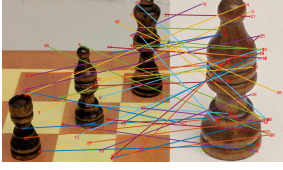


Figure 1: The SIFT features matching for the bishop. 2010 and 1211 SIFT features are extracted from left and right images respectively but only 40 matched features pairs are found.

expensive and not portable. Fortunately, with the rapid increase of computing power on mobile devices, an opportunity exists to apply computer vision methods to chess piece recognition, which is inexpensive and transferable.

Conventional approaches to object detection extract and match features such as the histogram of oriented gradient (HOG) [10] and the scale invariant feature transform (SIFT) [11]. These techniques work well when the objects have adequate visual texture. However, as shown in Fig. 1, very few effective SIFT features can be extracted from the small chess pieces since they do not have much distinguishable textures. Moreover, similar features among pieces complicate the matching process. In order to avoid incorrect matching, [12] and [13] assume the initial positions of the pieces are known, and then track the movement of pieces on the chessboard. However, those assumptions are undesirable and we want as few manual operations as possible.

Fortunately, although there is not much distinctive texture on the pieces, the different pieces have distinctive contours. A contour-based recognition method can match the observed contour to a template contour that is obtained from a model of the piece, or from a training image. By exploiting the relative positions of the edge points and normalizing the magnitudes, contour-based descriptors can be scale and rotation invariant like the Fourier descriptor with different shape signatures [14] and the context shape [15]. However, they also face some challenges. Methods using Fourier descriptors or polygonal approximations [16] may be affected severely when pieces have similar shapes or when occlusion occurs. A contour based method that is more robust to these effects is oriented chamfer matching [17, 18], and this is the method we selected.

Besides the above methods, convolutional neural networks have recently achieved great success in image classification and object detection problems [19, 20, 21], on large scale data sets like the ImageNet [22]. Therefore, we also implement several convolutional neural networks and compare them to our oriented chamfer matching approach. As far as we know, this is the first paper applying a convolutional neural network approach to the problem of 3D chess piece recognition under game conditions.

3. Chessboard Recognition

Chessboard recognition is an important first step towards piece recognition, since finding the board constrains the search for pieces. Additionally, we need to find the board in order to determine the relative locations of the pieces with respect to the board. As stated in the introduction, there are many chessboard recognition algorithms but only a few consider populated boards where the pieces cause occlusion. We chose to use a line detection based method since it is rare that a board line is completely occluded by the pieces. Specifically, we use the algorithm of [8] which achieves a high chessboard recognition success rate and more importantly, their workable viewing angles range covers the angles that a player would naturally look at the chessboard during a game. We briefly introduce their algorithm as follows.

Given a chessboard image, the Canny edge detector and Hough transform are used to find all possible lines in the image. The detected lines are clustered into two groups based on their locations in a scaled Hough transform space. These two groups correspond to the two orthogonal sets of lines on the chessboard. In the same space, outlier lines are filtered out by observing the relation between the detected lines. The intersections of two groups of remaining lines are calculated and recorded. Finally, all possible chessboard location candidates are transformed and matched to a chessboard reference model. The location with largest number of correct matching corners and the smallest matching residual error becomes the system output.

Once the chessboard lines are found, we need to find the pose of the board with respect to the camera, in order to predict the possible locations and appearance of the chess pieces. This requires the camera intrinsic parameter matrix K , and the board-to-camera rotation matrix R_B^C (R is used to indicate R_B^C in the following content). These two matrices can be estimated from the vanishing points of the two sets of chessboard lines by solving the following equations [23].

$$R_x = K^{-1} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \quad R_y = K^{-1} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} \quad (1)$$

$$K^{-1} = \begin{bmatrix} \frac{1}{f} & 0 & -\frac{c_x}{f} \\ 0 & \frac{1}{f} & -\frac{c_y}{f} \\ 0 & 0 & 1 \end{bmatrix} \quad (2)$$

$$\langle R_x, R_y \rangle = 0 \quad (3)$$

where R_x and R_y are the board coordinate system's bases in x and y directions. (x_1, y_1) and (x_2, y_2) are the vanishing points coordinates on the image plane. In addition, c_x , c_y and f are the optical center of the image and the camera focal length in pixels. Finally, the last column of the rotation

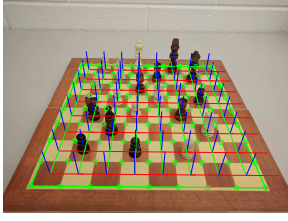


Figure 2: The chessboard preprocessing result. The board boundaries are marked by green lines and the normal vector of each square is indicated using a blue stick.

matrix, R_z , can be obtained by taking cross product of R_x and R_y .

Since we only have a single image of the chessboard, unless we know the size of the chessboard, there is no way to find out the true object scale. Therefore, we define a hyperplane using the board coordinate system's R_z basis as the support vector and a fixed constant to control the scale factor automatically. Based on the hyperplane and the rotation matrix, the normal vector for each square can be calculated and printed on the image as shown in Fig. 2 using blue sticks.

4. Piece Recognition

Once the pose of the chessboard has been found, the pose of each square can be estimated. This is needed to rotate and scale the templates that are used for matching. We will focus on piece recognition in the following sections.

4.1. Piece Location & Color Detection

Before matching templates, we want to determine possible piece locations in order to reduce the computation complexity. By leveraging the four chessboard corners in a homography transformation, an orthophoto (i.e., top-down view) of the chessboard is generated as shown in Fig. 3. Possible squares where pieces might be located are determined by counting the number of edge points in the areas that are indicated by green rectangles. An eight times eight matrix stores the possible squares occupied by pieces.

When the board is viewed from a very low angle, one chess piece might occupy several squares in the orthophoto like the bishop in Fig. 3 which covers both the square it occupies and the square behind it. In this case, a false indication of occupancy may occur. So a chamfer matching score threshold operation is implemented to avoid a false positive detection.

We next locate areas of interest (AOI) in the original image that may contain chess pieces. The size of an AOI in the image is relative to the viewing angle of the board. When the chessboard image is taken from a relatively low angle,

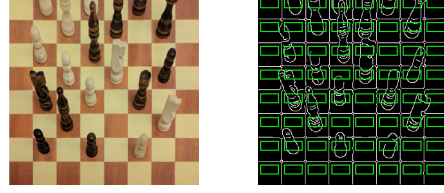


Figure 3: Left: Orthophoto of the board. Right: Search regions for occupied squares.

pieces are taller than in a direct overhead view. So a lower viewing angle leads to a larger AOI height. The height of the AOI must be large enough to contain the image of the largest pieces, which are the king and queen. The width of the AOI is set to the width of the corresponding square on the board.

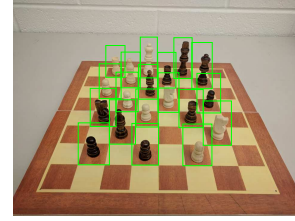


Figure 4: The AOIs in the input image.

We can determine the color of the pieces at this stage as well. Since we know the locations of the squares, we can find the average intensities for both black, I_b , and white squares, I_w . Each candidate's color is initiated to the square color which it stands on. The final decision can be easily made by comparing each candidate square's intensity, I_{ij} , to I_w and I_b .

$$P_{ij} = \begin{cases} \text{Black,} & \text{if } I_{ij} < k_w I_w, \text{ square } (i, j) \text{ is white} \\ \text{White,} & \text{if } I_{ij} > k_b I_b, \text{ square } (i, j) \text{ is black} \\ \text{same as the } (i, j) \text{ square's color} & \end{cases} \quad (4)$$

where P_{ij} indicates the color of the piece associated with the (i, j) square on the chessboard. k_w and k_b are scaling factors and in our project, $k_w = 0.7$ and $k_b = 1$.

4.2. Template Preparation

Three steps are performed in preparing the templates for matching. First, selecting the template based on the viewing angle. Second, rotating the template based on the normal vector. Third, scaling the template based on the square size.

For each chess piece, 12 templates with different viewing angles are captured as shown in Fig. 5. They range

from 10 to 70 degrees, where the template viewing angle is defined in Fig. 6. Note that the knight is not symmetrical around its vertical axis, so additional templates are needed for this piece to represent its appearance for rotations about the vertical axis. However, for simplicity, we assume all the knights are facing right and therefore only 12 templates are applied in this paper. During recognition, the viewing angle of the square being examined is calculated, and the templates nearest to that angle will be selected for the following translation and matching.



Figure 5: The bishop templates for chamfer matching.

Furthermore, the pieces do not always lie vertically and have varying sizes in the images due to their positions with respect to the camera. In the case that a piece is not vertical in the input image, we will rotate the templates accordingly as shown in Fig. 7 and scale it to fit into the observing square.

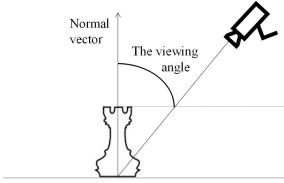


Figure 6: The viewing angle.

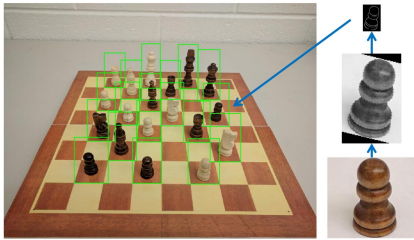


Figure 7: The selected and translated pawn's template.

4.3. Oriented Chamfer Matching

As previously stated, we use a contour-based recognition method because of the lack of texture features. Chamfer distance matching, originally proposed in [24], is a well-established contour matching technique which measures

the similarity between the objects in the input image and templates. For every candidate object position, a chamfer matching score is calculated. The object's class and location are determined by the template and the region that get the minimum chamfer matching score.

The traditional chamfer matching requires the edge images for both the input image, I , and the template, T . The chamfer distance can be obtained by solving the following least square problem where $|T|$ is the number of total edge points in the template and τ is the truncation parameter for normalization. In our project, $\tau = 30$.

$$d_{dist}(x) = \frac{1}{\tau|T|} \sum_{x_t \in T} \min(\tau, \min_{x_i \in I} \|(x_t + x) - x_i\|_2). \quad (5)$$

For a specific matching starting point x in the input image, the chamfer distance score is the average distance between the template edge points and their nearest edge points in the input image. Furthermore, the above least square problem can be solved efficiently by mapping the desired template's edge image onto a pre-computed input image's distance transformation image and summing up the element-wise product of pixel intensities within the template covered region.

To provide additional stability and resistance to background noise, edge orientation is adopted to compare the gradient differences [17, 18]. The orientation score can be calculated by solving the following least square problem where ϕ is a function measuring the edge point's orientation in radians. The physical meaning of ϕ and d_{dist} in the input image can be found in Fig. 8.

$$d_{orient}(x) = \frac{2}{\pi|T|} \sum_{x_t \in T} |\phi(x_t) - \phi(\arg \min_{x_i \in I} \|(x_t + x) - x_i\|_2)|. \quad (6)$$

Similarly, the orientation score can also be calculated efficiently using the pre-computed gradient images. The final chamfer score is calculated by:

$$d_{score}(x) = (1 - \lambda)d_{dist}(x) + \lambda d_{orient}(x), \quad (7)$$

where λ is a weight factor in the range of $[0, 1]$. In our project, $\lambda = 0.5$ and the detailed analysis regarding different values of λ can be found in the section 5.6. A perfect matching would get a score of 0. After template matching, the template with smallest oriented chamfer matching score and its corresponding location will be marked on the input image for each AOI. Templates with high scores are rejected.

4.4. Matching Process

The matching process is quite straight forward. For each AOI, all templates taken from the angle that matches the

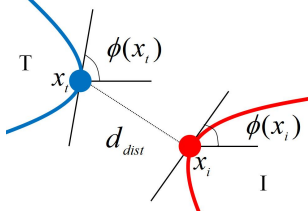


Figure 8: The oriented chamfer matching.

observing square’s viewing angle are selected and translated for chamfer matching. A list stores the chamfer scores for all different templates and records the template with the minimum score. In addition, to expedite the matching process, an N -sampling strategy is applied. Namely, we compute the chamfer score with a stride of N pixels if we are in a high score area, but compute the score at every pixel in the low score areas. The idea is to focus our computational resources on the most promising piece locations. After finishing all AOI matching, the recognition results including the pieces colors, names and their corresponding locations are shown on the input image as shown in Fig. 9.

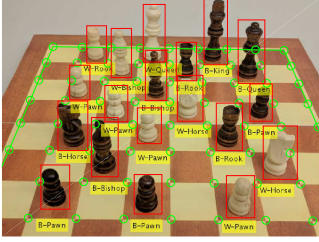


Figure 9: The recognition result.

We can reject invalid piece detections by a threshold on the chamfer matching score. To determine this threshold, we recorded the oriented chamfer matching scores for different templates and true classes for a typical image in Table 1. Based on the table, 0.2 is a reasonable threshold to rule out a false positive detection.

5. Experiments

We tested our approach and compared it to several alternative approaches based on convolutional neural networks, on a series of real chessboards taken from varying angles and different resolutions. In addition, we quantify the effect of occlusion and pan angles and evaluate their processing time. Furthermore, we study the performance with different algorithm parameters. Examples of input images and the recognition results are shown in Table 2.

5.1. Experimental Setup

In order to imitate the views that a player would naturally have during a real game, the viewing angle of the test images is approximately 40 degrees using the definition in Fig. 6. The sampling mode is 3-sampling and $\lambda = 0.5$. Thirty test images are taken and the number of pieces by type is shown in Table 3.

Table 3: The pieces distribution of the test set.

Board	King	Queen	Bishop	Knight	Rook	Pawn
30	43	32	76	63	98	173

In addition, several test sets with same piece distribution but different occlusion conditions and pan angles are collected. In all test sets, we assume there is no piece directly behind another since we will study the effect of occlusion individually.

5.2. Convolutional Neural Networks

In this experiment, we selected three of the most popular convolutional neural networks, GoogleNet [19], ResNet [20] and VGG [21], to compare with the oriented chamfer matching approach. Furthermore, the research of transfer learning shows that the learned CNN features are transferable among similar tasks [25]. Therefore, all the selected networks are pre-trained on the ImageNet [22] classification data set for initialization. And to adapt to the piece recognition application, the networks’ last layers are replaced by a softmax regression with six output nodes and all test images are resized to $223 \times 223 \times 3$ pixels accordingly. The Adam optimization algorithm [26] is applied with 0.001 learning rate and 1000 maximum iteration number. To train the system, we took 20 additional chessboard images and extracted the pieces as the training set which contains pieces images with varying viewing angles and colors. The number of training images for each piece type is listed in Table 4 and four bishop training examples are shown in Fig. 10.

Table 4: The number of training images of each piece type for convolutional neural networks and oriented chamfer matching.

Convolutional neural network					
King	Queen	Bishop	Knight	Rook	Pawn
40	40	40	40	40	60
Oriented chamfer matching					
King	Queen	Bishop	Knight	Rook	Pawn
12	12	12	12	12	12

In the first experiment, we train and evaluate the neural networks and oriented chamfer matching’s performance on images where the pan angle of the camera (the rotation about the vertical axis) with respect to the board is zero degrees. Pieces have less than 10% occlusion and the resolu-

Table 1: The oriented chamfer matching scores.

True class \ Template	King	Queen	Bishop	Knight	Rook	Pawn
King	0.1285	0.1705	0.2201	0.2041	0.1930	0.2050
Queen	0.1537	0.0605	0.1969	0.1731	0.1674	0.2016
Bishop	0.3044	0.3482	0.0764	0.2007	0.1270	0.1669
Knight	0.3283	0.3473	0.2550	0.0925	0.1820	0.1868
Rook	0.1992	0.1838	0.1288	0.1871	0.0860	0.1389
Pawn	0.2809	0.2701	0.1899	0.2605	0.1994	0.0794
Empty square	0.3083	0.2619	0.2754	0.2778	0.2588	0.2754

Table 2: The 3D chess pieces recognition experiments. The first row shows the recognition process of a 720×960 pixels test image. The second row shows the recognition process with a 240×320 pixels test image. The third row shows the 60% occlusion image's recognition process and the last row shows the recognition process on a test image with a 30 degree pan angle.


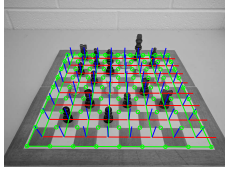
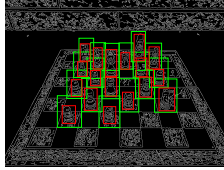


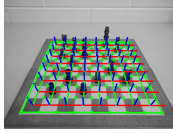
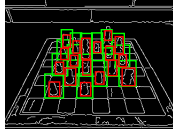
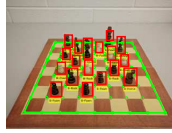

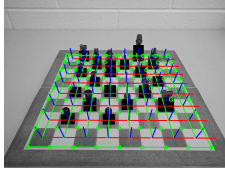
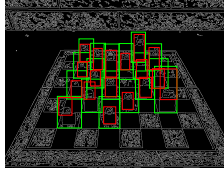
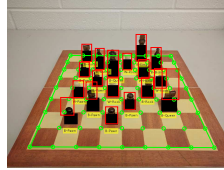

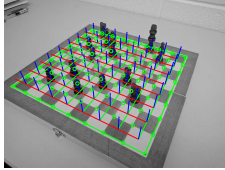
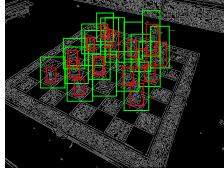
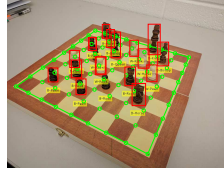
Input images	Preprocessing	Templates matching	Recognition result
			
			
			
			



Figure 10: Four bishop training examples for CNN.

tion of the images is 720×960 pixels. Their recognition accuracy is recorded in Table 5 from which we can observe that all approaches perform quite well at piece recognition. The oriented chamfer matching method achieves 95.46%

accuracy which is better than ResNet50 but slightly worse than GoogleNet and VGG-16. However, to achieve this performance, the neural networks require 3.6 times larger training set than the oriented chamfer matching.

5.3. Effect of Resolution

In this section, we evaluate the effect of image resolution. We use 120, 240, 360, 480 and 720 to indicate 120×160 , 240×320 , 360×480 , 480×640 and 720×960 resolution test sets respectively and record both the convolutional neural networks and the oriented chamfer matching's

Table 5: The recognition accuracy for different approaches.

	King	Queen	Bishop	Knight	Rook	Pawn	Overall
GoogleNet	97.67%	100.00%	100.00%	100.00%	97.96%	96.53%	98.14%
VGG-16	100.00%	90.63%	97.37%	98.41%	87.76%	99.42%	96.08%
ResNet50	88.37%	100.00%	100.00%	100.00%	81.63%	97.69%	94.43%
Oriented Chamfer	90.70%	90.63%	85.53%	100.00%	95.92%	100.00%	95.46%

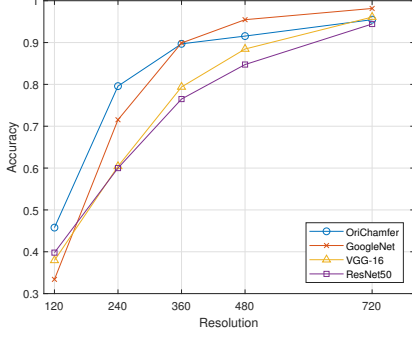


Figure 11: The recognition accuracy with different resolutions.

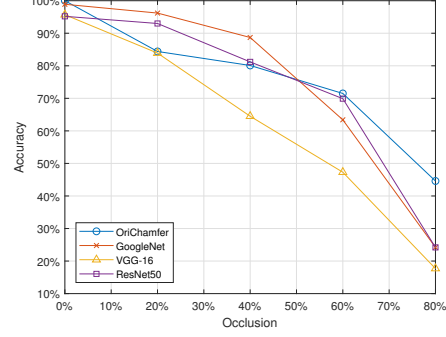


Figure 12: The recognition accuracy with different percentages of occlusion.

overall recognition accuracy in Fig. 11.

The oriented chamfer matching outperforms convolutional neural networks when the images are taken by a low resolution camera. It may be that the low resolution test images lose the features that neural networks learned from the high resolution training images.

5.4. Effect of Occlusion and Pan Angle

The above two experiments are evaluated on the test set with no or slight occlusion ($< 10\%$ occlusion). To quantify the occlusion effect, we select several test images where all pieces are successfully recognized and start occluding the pieces with a 10% interval. Specifically, 60% occlusion means 60% area of the pieces from the bottom is occluded and an example is shown in the 3rd row in Table 2. The overall accuracy for both convolutional neural networks and oriented chamfer matching under different occlusion conditions is recorded in Fig. 12. As expected, accuracy decreases as the occlusion effect becomes stronger. We observe that under severe occlusion ($\geq 60\%$), oriented chamfer matching outperforms the convolutional neural networks. It is possible that the convolutional neural networks might perform better in these cases if the training set included many more examples of occluded pieces.

Finally, in a real usage scenario, the camera may pan around the chessboard. Therefore, we also evaluate the approaches with different pan angles in Fig. 13. It can be observed that panning the camera away from the zero angle brings down the accuracy. The oriented chamfer matching achieves similar accuracy to GoogleNet while outperforms

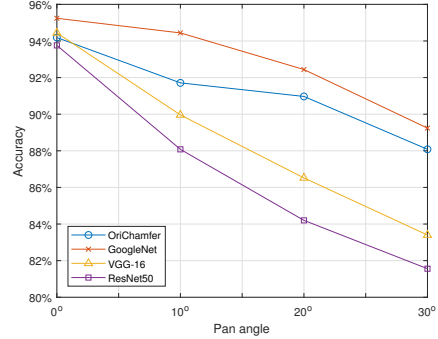


Figure 13: The recognition accuracy with different pan angles.

the VGG-16 and ResNet50.

5.5. Processing Time

Regarding the efficiency, we evaluate the convolutional neural networks and oriented chamfer matching in terms of the processing time and they are implemented using TensorFlow [27] and Matlab respectively on an i7 6700K CPU. For the oriented chamfer matching, two major factors affecting the processing time are the sampling method and the image resolution. By manipulating these two factors, we acquire the average processing time of oriented chamfer matching for different settings in Table 6. Lower resolution implies smaller searching area and the same applies for the sampling method. The convolutional neural networks' testing time is also recorded in Table 6. We find that if we choose

Table 6: The processing (testing) time for recognizing 10 pieces (unit: second). Different resolution images should lead similar testing time for neural networks since after preprocessing, all images would have the same dimension.

Oriented Chamfer	120	240	360	480	720
0-Sampling	1.3776	1.4578	1.9259	3.4092	7.2975
3-Sampling	1.3932	1.4472	1.8124	3.0181	5.3851
6-Sampling	1.3975	1.3809	1.7419	2.8735	4.7469
9-Sampling	1.3796	1.3772	1.6405	2.7951	4.3742
12-Sampling	1.3719	1.3723	1.6662	2.6239	4.1666

Network	Time
GoogleNet	1.2181
VGG-16	8.2453
ResNet50	3.1680

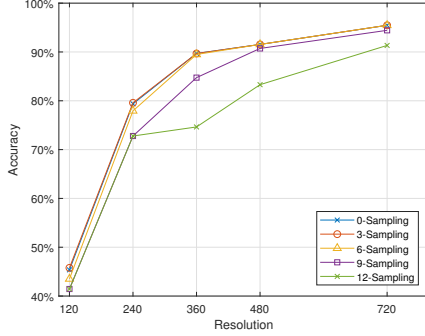


Figure 14: The recognition accuracy with different sampling methods and resolutions.

9-sampling method for the 720 resolution test set, the oriented chamfer matching has comparable processing time to the neural networks.

In addition, there is a tradeoff between processing time and accuracy for oriented chamfer matching. To visualize the tradeoff, we evaluate the overall accuracy for different settings in Fig. 14. In the low resolution, the width of each piece is too short to capture useful edge structures and the 12-sampling method might skip the ground true locations. Both cases lead very low overall accuracy.

5.6. Lambda

Another important factor in the oriented chamfer matching is the parameter λ , which controls the weighting of the distance score to the orientation score. When $\lambda = 0$, the oriented chamfer matching degenerates to the chamfer distance matching [24]. When $\lambda = 1$, only the orientation term is applied. We examine and record the overall accuracy with different λ in Fig. 15. The accuracy with zero λ is far smaller than other settings. Because in a noisy edge image, the distortion of the templates combining with the false edge points may lead the false matching while the orientation term provides an effective guideline to rule out this situation. In addition, $\lambda = 0.5$ achieves the highest accuracy in most cases which makes it an excellent choice for pieces recognition.

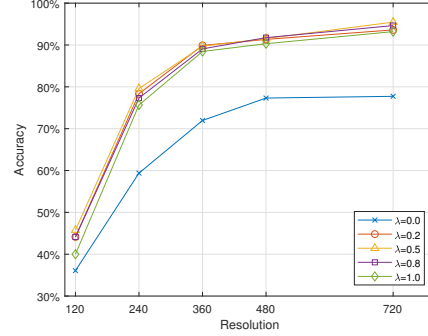


Figure 15: The recognition accuracy with different values of λ .

6. Conclusion

In this paper, we present an approach for 3D chess piece recognition using oriented chamfer matching. After recognizing the chessboard, we can select the appropriate templates for matching and compute the oriented chamfer score efficiently. We quantify the effect of resolution, occlusion and pan angles, analyze the processing time and accuracy tradeoff and examine the effect of different algorithm parameters. We also implement the convolutional neural networks for comparison. In experiments, the chamfer matching approach achieves similar performance as the convolutional neural networks, but uses a much smaller training set and avoids the time consuming training process. In addition, the oriented chamfer matching is more robust in severe occlusion and low resolution cases. This result may follow from the fact that in the chamfer matching method, we explicitly give the system information on what features belong to the object, but in the convolutional neural networks, the system must learn what is object versus background from training examples. It is possible that if more training examples were used, the performance of the convolutional neural networks might improve in severe occlusion and low resolution cases. However, the collection of labeled training images is time consuming and a burden for the user. Since the performance of the two approaches is otherwise comparable, this might indicate the choice of the oriented chamfer matching approach.

References

- [1] H. Kaufmann, "Collaborative augmented reality in education," *Institute of Software Technology and Interactive Systems, Vienna University of Technology*, 2003.
- [2] R. K. Miyake, H. D. Zeman, F. H. Duarte, R. Kikuchi, E. Rammacciotti, G. Lovhoiden, and C. Vrancken, "Vein imaging: a new method of near infrared imaging, where a processed image is projected onto the skin for the enhancement of vein treatment," *Dermatologic surgery*, vol. 32, no. 8, pp. 1031–1038, 2006.
- [3] T. Hodan, P. Haluza, Š. Obdržálek, J. Matas, M. Lourakis, and X. Zabulis, "T-less: An rgb-d dataset for 6d pose estimation of texture-less objects," in *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on*, pp. 880–888, IEEE, 2017.
- [4] A. Geiger, F. Moosmann, Ö. Car, and B. Schuster, "Automatic camera and range sensor calibration using a single shot," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 3936–3943, IEEE, 2012.
- [5] A. De la Escalera and J. M. Armingol, "Automatic chessboard detection for intrinsic and extrinsic camera parameter calibration," *Sensors*, vol. 10, no. 3, pp. 2027–2044, 2010.
- [6] J. E. Neufeld and T. S. Hall, "Probabilistic location of a populated chessboard using computer vision," in *Circuits and Systems (MWSCAS), 2010 53rd IEEE International Midwest Symposium on*, pp. 616–619, IEEE, 2010.
- [7] K. Y. Tam, J. A. Lay, and D. Levy, "Automatic grid segmentation of populated chessboard taken at a lower angle view," in *Computing: Techniques and Applications, 2008. DICTA'08. Digital Image*, pp. 294–299, IEEE, 2008.
- [8] Y. Xie, G. Tang, and W. Hoff, "Geometry-based populated chessboard recognition," in *Tenth International Conference on Machine Vision (ICMV 2017)*, International Society for Optics and Photonics, 2017.
- [9] L. Miolo, "Magnetic chessboard with self-centering pieces," Nov. 10 1981. US Patent 4,299,389.
- [10] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, pp. 886–893, IEEE, 2005.
- [11] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.
- [12] T. Cour, R. Lauranson, and M. Vachette, "Autonomous chess-playing robot," *Ecole Polytechnique*, July, 2002.
- [13] N. Banerjee, D. Saha, A. Singh, and G. Sanyal, "A simple autonomous robotic manipulator for playing chess against any opponent in real time," in *Proceedings of the International Conference on Computational Vision and Robotics*, 2011.
- [14] M. Yang, K. Kpalma, and J. Ronsin, "Shape-based invariant feature extraction for object recognition," *Advances in Reasoning-Based Image Processing Intelligent Systems*, pp. 255–314, 2012.
- [15] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE transactions on pattern analysis and machine intelligence*, vol. 24, no. 4, pp. 509–522, 2002.
- [16] H. Imai and M. Iri, "Polygonal approximations of a curve," *Computational Morphology*, pp. 71–86, 2014.
- [17] J. Shotton, A. Blake, and R. Cipolla, "Multiscale categorical object recognition using contour fragments," *IEEE transactions on pattern analysis and machine intelligence*, vol. 30, no. 7, pp. 1270–1281, 2008.
- [18] A. Thayananthan, B. Stenger, P. H. Torr, and R. Cipolla, "Shape context and chamfer matching in cluttered scenes," in *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, vol. 1, pp. I–I, IEEE, 2003.
- [19] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.
- [20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.
- [21] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.
- [22] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al., "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.
- [23] R. Szeliski, *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.
- [24] H. G. Barrow, J. M. Tenenbaum, R. C. Bolles, and H. C. Wolf, "Parametric correspondence and chamfer matching: Two new techniques for image matching," tech. rep., DTIC Document, 1977.
- [25] A. Sharif Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "Cnn features off-the-shelf: an astounding baseline for recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 806–813, 2014.
- [26] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.
- [27] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.