Rubik's Cube Reconstruction from Single View for Service Robots

Włodzimierz Kasprzak, Wojciech Szynkiewicz, and Łukasz Czajka
Warsaw University of Technology, Institute of Control and Computation Eng.
ul. Nowowiejska 15/19, PL-00-665 Warsaw
e-mail: W.Kasprzak@ia.pw.edu.pl

Abstract. Rubik's cube puzzle is considered to be a benchmark for service robots. In such an application a computer vision subsystem is required in order to locate the object in space and to determine the configuration of its color cells. In this paper a robust algorithm for Rubik's cube reconstruction from a single view in real time is presented. Special interest is in getting a balance between quality of results and computational complexity of the algorithm.

Key words: color analysis, shape description, segment grouping, service robots, surface orientation, 3-D object recognition

1. Introduction

Quite recently the Rubik's cube puzzle emerged as a benchmark for cooperating service robots [2], [7], [8]. Typical solutions require the use of a computer vision subsystem that processes images from one or more cameras. Main goals of image processing in this context are [8]: (1) an initial detection of the cube, (2) the 3-D reconstruction of the Rubik's cube pose (position and orientation), which enables the manipulator to grasp it, and finally (3) an appropriate recognition of the color cells of all the 3 visible surfaces of the cube.

The reconstruction of a 3-D object in computer vision generally requires the extraction of image features in several views, detection of correspondences and (approximately, in terms of the smallest total error) the solution of a system of equations [1]. An obvious observation is that the use of geometric constraints can simplify the reconstruction problem, e.g. planarity, alignments, symmetries, known angles between directions [3], [5]. As the Rubik's cube is of well-defined structure we can use a single view and combine it with prior knowledge, i.e. the number of cells and their colors [4].

In this paper an algorithm for Rubik's cube reconstruction from a single view in nearly real time is shown. Focus was on getting appropriate balance between quality of results and computational complexity of the algorithm. Section 2 provides an introduction to the application domain, i.e. a service robot benchmark, and it gives the overall solution structure. In section 3 the image analysis process is described: color processing, cell detection, cell grouping and surface reconstruction.

2. Benchmark for a service robot

The development of service/personal robots functioning in a human-oriented environment is a major scientific and technological challenge. One of the challenging problems for service robots is grasping and manipulating objects in a semi-structured environment, where object properties are not known *a priori* and sensing is prone to errors. Rubik's cube puzzle solution is a task requiring nearly all skills that are essential in a service robot. Thus, Rubik's cube manipulation is an example of highly demanding benchmark task for two-handed service robots, because it requires from the system:

- two-handed manipulation to efficiently turn the faces of the cube;
- visual capabilities to locate the cube;
- visual servo-control to get hold of the cube;
- position-force control to avoid jamming of the cube while rotating the faces;
- fusion of deliberative and behavioral control to work out the plan of motions solving the puzzle and to adapt quickly to sudden changes in the environment (e.g. in the case of jamming).

Hence, a vision subsystem is responsible both for detecting and locating the cube in 3-D space, and identifying its initial and current configuration, i.e. color cell configuration within the faces of the cube.

The overall experimental setup consists of two 6 degree of freedom (dof) modified IRp-6 robot arms, each with a parallel jaw gripper (Fig. 1). Each jaw is instrumented with tactile sensors which detect the presence of contacts with the grasped object. Moreover, each hand is equipped with a wrist-mounted ATI Gamma force-torque sensor, and an eye-in-hand miniature CCD color camera. Additionally, a global vision system with fixed-mount SVS-Vistek SVS084 CCD color camera and Leonardo Digital Video Processor from Arvoo for fast image acquisition and realtime processing of the incoming data is used.

3. Rubik's cube reconstruction

The image analysis procedure enables the detection of a Rubik's cube, based on a previous detection of color cells and their grouping into up to 3 surfaces (Fig. 2).

3.1. Color-based iconic processing

Proper color-based image processing requires that the white level in the input image is balanced, i.e. that all levels of grey differ only by their intensity component Y, whereas their color components U and V are centered. The input image is transformed from the RGB space into the YUV space. Then the pixel color components are normalized with

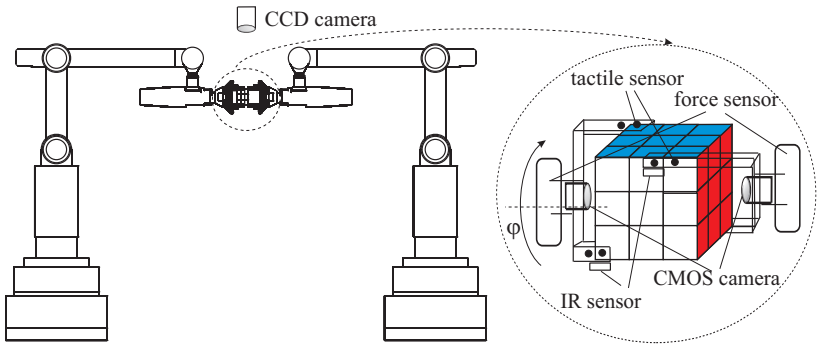


Fig. 1. Sensors used to locate and manipulate the Rubik's cube.

Fig. 2. The input image and the final segmentation image that contains cell groups (surfaces).

respect to pixel's intensity (Fig. 3):

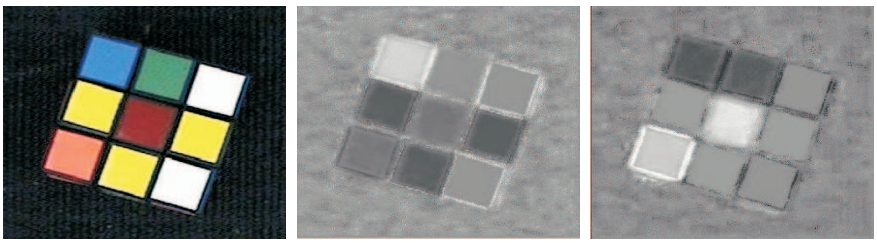
$$Y'_p = 128; U'_p = (U_p - 128) \cdot \kappa_p + 128; \tag{1}$$

$$V'_p = (V_p - 128) \cdot \kappa_p + 128; \tag{2}$$

where

$$\kappa_p = \begin{cases} \frac{Y'_p}{Y_p} & \text{for } Y_p > 128 \\ \frac{256 - Y_p}{Y'_p} & \text{for } Y_p < 128 \\ 1 & \text{for } Y_p = 128 \end{cases} \tag{3}$$

Eventually an overflow can appear, i.e. $U'_p, V'_p > 255$, but this is handled separately as overflow is caused only by "dark" or "light" pixel. This normalization improves significantly the spiky-ness of the histogram function - although the background pixels are still dominant in the histogram, but now there also appear clearly other local maximum peaks (Fig. 4).



(a) (b) (c)

Fig. 3. The input image (a) and the corresponding U- and V-images (b, c) (after Y-based normalization).

Fig. 4. The histograms of normalized U- and V-images.

Next pixel classification into hypothetic cell colors can be done (Tab. 3.1). A precondition for color detection, besides the color components U'_p and V'_p of every pixel p, is sufficient (original) intensity Y_p, i.e. a very "dark" or very "white" pixel cannot be Y-normalized and it contains no valuable color information.

Please note that this approach requires a previous color-based lighting calibration.

Tab. 1. The color component intervals for color detection under "perfect white" lighting conditions.

$Color_p$	Y_p	U'_p	V'_p
	min-max	$\mu, 3\sigma$	$\mu, 3\sigma$
red	50-140	110, ±38	205, ±45
orange	140-210	100, ±48	205, ±45
blue	70-240	195, ±55	55, ±55
green	70-240	70, ±65	60, ±50
white	190-255	128, ±12	128, ±11
yellow	118-150	90, ±25	140, ±20

In perfect "white" lighting conditions the U and V components of a "white" pixel are located in the scale center. Accordingly the intervals for specific colors are observed and stored. If the lighting system changes, such that the "white"-level moves nearer to green or further away from green, then the other color intervals change appropriately. We need to know the current U and V components of the "white" color in order to compensate the distance from perfect lighting conditions. This compensation works fine for restricted deterioration only of the lighting color.

3.2. Cell recognition

The color-filtered image constitutes the input for cell detection (Fig. 5). The color image is post-processed by a subsequent erosion- and dilation-like filtering. This separates well the color regions one from the other, while fully "flooding" the internal homogeneous regions. Then a border tracking of homogeneous regions is performed.

(a) (b) (c)

Fig. 5. The steps of image segmentation in the ROI region: (a) after color-based filtering, (b) after erosion, (c) after dilation.

The shapes of detected borders are characterized next. Several functions based on geometric moments represent the invariant features of closed contours and by assigning appropriate constraints to them the contours of interest (corresponding to Rubik's cube cells) can be detected. Among the 10 tested moment-based functions, the following four functions with corresponding conditions, that are suitable for rectangular shapes, were selected: $M1 \geq 1.0$, $M2 \leq 2.5$, $M3 \leq 0.5$ and $M7 \geq 0.1$. They were experimentally discovered to perform best with respect to our task:

$$M1 = (\mu_{20} + \mu_{02})/m_{00}^2 \tag{4}$$

$$M2 = \left[(\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2\right]/m_{00}^4 \tag{5}$$

$$M3 = \left[(\mu_{30} - 3\mu_{12})^2 + (3\mu_{21} - \mu_{03})^2\right]/m_{00}^5 \tag{6}$$

$$M7 = (\mu_{20}\mu_{02} - \mu_{11}^2)/m_{00}^4 \tag{7}$$

where m_{pq}, μ_{pq} are the global and central moments of order (p,q), with respect to the first and second coordinate.

Unfortunately, large variability of proper rectangular contours, which are projections of the square-like faces from different viewpoints, can be observed. Hence, in order not to reject any proper rectangle, we have to keep extended interval conditions for above four moment functions. This causes that other non-proper rectangular contours also pass these tests. To distinguish proper from non-proper rectangular contours we finally check the normalized compactness coefficient γ of every contour that passed the moment-based tests:

$$\gamma = \frac{L}{2\sqrt{\pi S}} - 1 \tag{8}$$

which should be in the range of: $\gamma \in [0, 0.4]$.

3.3. Cell configuration and surface detection

The surface detection step starts with the computation of 4 vertices for every contour - this leads to 2 main directions of every enclosed cell. The topological information (neighborhood) and the available directions allow the grouping of cells into surfaces (Fig. 6). A more realistic case with 3 visible surfaces is shown in Fig. 7.

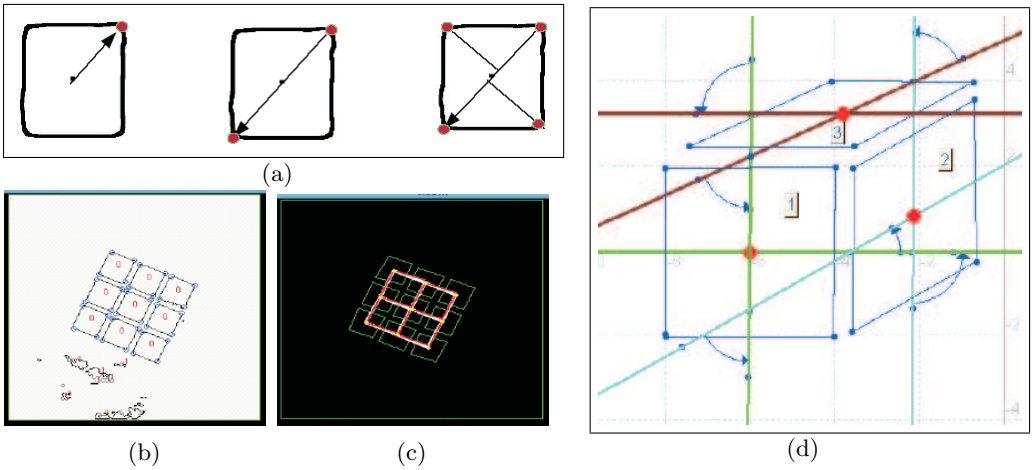
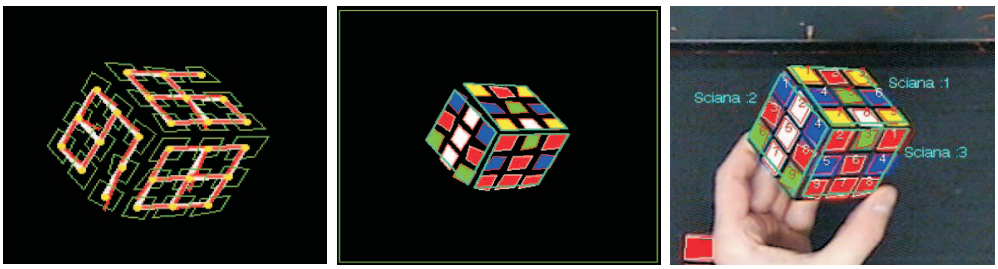


Fig. 6. The steps of surface detection: (a) detection of 4 vertices, (b) cell border approximation (vertices and border lines), (c) the topological structure (neighborhood) of cells, (d) the directional information, based on vertex pairs, when 3 surfaces are visible.



(a) (b) (c)

Fig. 7. The Rubik's cube description steps when three surfaces are visible: (a) the topological information for 3 visible surfaces, (b) the detected surfaces, (c) the final Rubik'c cube description.

3.4. 3-D object reconstruction

The availability of camera calibration parameters is important for 3-D reconstruction - then the 2-D point coordinates can be related to metric measurements. The Rubik's cube reconstruction problem can be decomposed into cell configuration recognition and the estimation of the object-to-camera coordinate system transformation. It is possible to detect the full cell configuration from a single view only, when 3 Rubik's cube faces are visible. Otherwise fusion of information coming from 3 different views is needed. This is out of the scope of this paper.

The transformation of a scene point $\mathbf{P}$ onto an image point $\mathbf{p}$ can be expressed by a 4×4 homogeneous coordinate transformation matrix:

$$\mathbf{p} = \begin{bmatrix} kx \\ ky \\ kz \\ k \end{bmatrix} = \mathbf{A} \cdot \mathbf{P} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ . & . & . & . \\ . & . & . & . \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{9}$$

Obviously $z = f$, where f is the focal length of the camera. The third row is always linearly dependent on the fourth row of matrix $\mathbf{A}$ - we can consider three equations corresponding to the 1-st, 2-nd and the 4th row. After elimination of the unknown coefficient k from such a system of equations we have 2 potentially independent equations:

$$a_{11}X + a_{12}Y + a_{13}Z - a_{41}xX - a_{42}xY - a_{43}xZ - a_{44}x + a_{14} = \epsilon_1 \tag{10}$$

$$a_{21}X + a_{22}Y + a_{23}Z - a_{41}yX - a_{42}yY - a_{43}yZ - a_{44}y + a_{24} = \epsilon_2 \tag{11}$$

where ϵ_i is the approximation error related to the i-th equation.

All the parameters a_{ij} of the above matrix $\mathbf{A}$ depend on the following unknown

parameters of the object-to-camera coordinate transformation: the translation vector $\mathbf{T} = [t_X, t_Y, t_Z]$, the rotation angle vector $\mathbf{\Theta} = [\alpha_Z, \beta_X, \gamma_Y]$.

With the knowledge of the focal length the depth of object's origin point can be estimated from the relation between the known real size of the object and its projected image size. All the corner points of the cube can be expressed in relation to this central point with the help of the 6 unknown transformation parameters. Even if only one surface is visible then we have 4 visible corners and thus a system of 8 equations can be defined. Hence all the unknown transformation parameters can be estimated. Obviously these 4 points are coplanar and only one rotation coefficient can be estimated from them, whereas the two remaining rotations are set to zero.

4. Tests

The implementation was tested on a computer with a Pentium IV, 2400 MHz processor. We distinguish two processing modes: (1) the initialization mode - for the first part of an image sequence, from the first image up to the image in which the Rubik's cube is detected - which requires the processing of the entire image, and (2) the tracking mode - for the second part of the sequence that follows an initial detection of the cube - which needs to process a restricted image area only.

We noticed the following average face detection times:

1. if the entire image is processed (initialization mode):
   (a) one face is in view - 90 ms;
   (b) more than one face is visible - 100 ms;
2. processing a ROI part of the image only (tracking mode):
   (a) one face is in view : 32 ms;
   (b) more than a single face is in view: 52 ms.

5. Summary

This work shows that a robust and fast reconstruction of the Rubik's cube is possible from a single view only. Thus the computer vision system can be used for the benchmark task solution defined for service robots, i.e. to solve the Rubik's puzzle by two cooperating manipulators.

Acknowledgments

The research work reported in this paper was supported by a grant of the Polish Ministry of Scientific Research and Information Technology - MNiI 4T11A 003 25.

References

1993
[1] O. Faugeras. *Three-dimensional computer vision. A geometric viewpoint.* The MIT Press. Cambridge, Mass. 1993.

1997
[2] R. Korf. Finding optimal solutions to Rubik's cube using pattern databases. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-97)*, July 1997, pp. 700–705.

1999
[3] P.F. Surm and S.J. Maybank. A method for interactive reconstruction of piecewise planar objects from single views. *British Machine Vision Conference, BMVC 1999*. BMVA Press 1999, pp. 265–274.

2000
[4] J. Bruce, T. Balch, and M. Veloso. Fast and inexpensive color image segmentation for interactive robots. In *Proc. of the IEEE/RSJ Int. Conference on Intelligent Robots and Systems (IROS'00)*, volume 3, 2000, pp. 2061–2066.

2004
[5] L. Venturino et al.. Improving 3D scene reconstruction through the application of geometric constraints. In *IWSSIP'04. Int. Workshop on Systems, Signals and Image Processing*, Poznan University of Technology Press, September 2004, pp. 115–118.

2005
[6] O. Tahri and F. Chaumette. Point-based and region-based image moment for visual servoing of planar objects. *IEEE Transactions on Robotics*, 21(6):1116–1127, Dec 2005.

2006
[7] W. Szynkiewicz, C. Zieliski, W. Czajewski and T. Winiarski. Control architecture for sensor-based two-handed manipulation. In T. Zieliska and C. Zieliski, editors. *CISM Courses and Lectures*, number 487, pp. 237–244, Springer, Wien-New York, 2006.
[8] C. Zieliski, W. Szynkiewicz, T. Winiarski, and M. Staniak. Rubik's cube puzzle as a benchmark for service robots. In *12th IEEE International Conference on Methods and Models in Automation and Robotics, MMAR'2006*, pp. 579–584, August 2006.