OBJECT RECOGNITION USING KINECT

ROHIT RAWAT

SHREYAS KASHYAP

ABSTRACT

We've seen SURF/SIFT features being used for matching objects from a database in a scene. Objects being matched always have a perspective transformation in the scene. For example, an image in our database might be a rectangular picture, but in the scene, it may appear in a trapezoidal shape. RANSAC is known to work around these homographic transforms. But since the image is already distorted, the SIFT descriptors that come out of it are not the best for matching with descriptors extracted from a planar image in our database. We think we can improve the accuracy and efficiency of matching by using depth information from a Kinect camera to remove or mitigate this perspective transformation. The Kinect camera will be able to provide us a 3D projection of a scene. We will first extract planar surfaces like walls, floors, ceilings and tables using Hough transforms or RANSAC. We will then transform the image sections to their orthographic views. We believe that SIFT matching done on this orthographic view will give better results. We will focus on planar object like pictures.

Introduction

As seen in the images below, SIFT works very well when matching two images taken from the same perspective.

(a) (b)

(c)

Figure 1. (a) An image to be matched (b) Cluttered scene with the image, (c) Detected location of the image.

But when the image is to be found in a scene observed from a different angle, the SIFT/RANSAC based matching can be very unreliable, for example in the following scene.

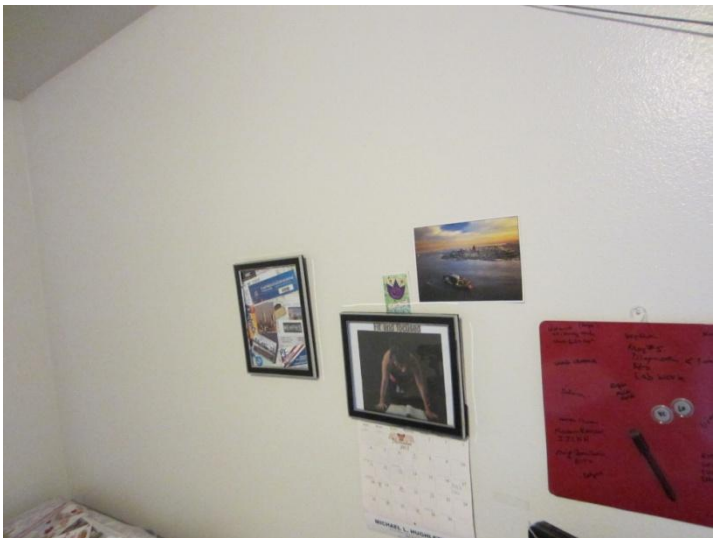


Figure 2. Scene of a wall viewed at an oblique angle.

Our goal in this project is to transform an image as in Fig. 2 to an orthogonal view using 3D information and homography.

WORKING STEPS

The steps that we followed during our project can be briefly divided into the following:

Use EGT to generate simulated data

We created a room with two walls and a table. We placed random 3D points on all three surfaces to give physical form to the scene. We also simulated three rectangular objects (pictures or books) on each surface by using a dense matrix of points. This 3D information would conventionally come directly from the Kinect camera. Random noise was added to the points to reflect the real world.

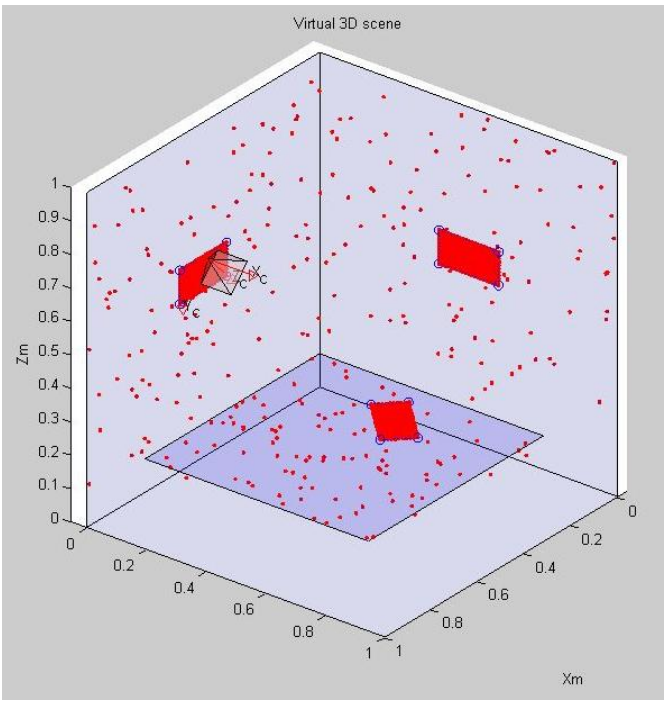


Fig 3. 3D scene of the room.

Using EGT's perspective projection function, we created the scene as seen by Kinect's RGB camera, shown in Fig 4.

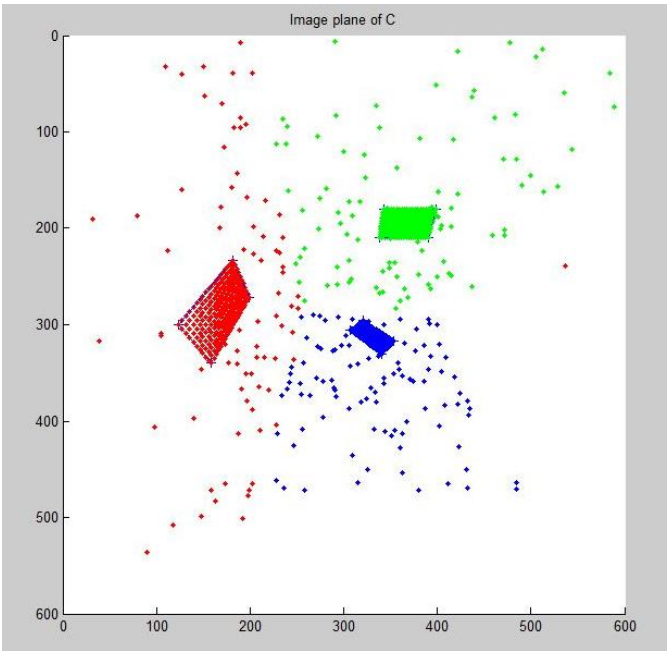


Fig. 4. Simulated view from the Kinect RGB camera.

Segment planes using the simulated data to get planes ∏(1), ∏(2), ∏(3).. etc.

We used the Hough transform to detect planes. The purpose of the technique is to find imperfect instances of objects within a certain class of shapes by a voting procedure. In our project, we have used this technique on the planes to extract the image features. We used two parameters in performing the hough transform

1. the distance from the origin
2. normals of the planes

We have used a matrix called normals for initializing the normals that have to be detected. For example if we want to extract the xy plane, then the value [0 0 1] is stored in the normal matrix to represent the xy plane. We have a matrix called distances which stores the distances of points from the origin.

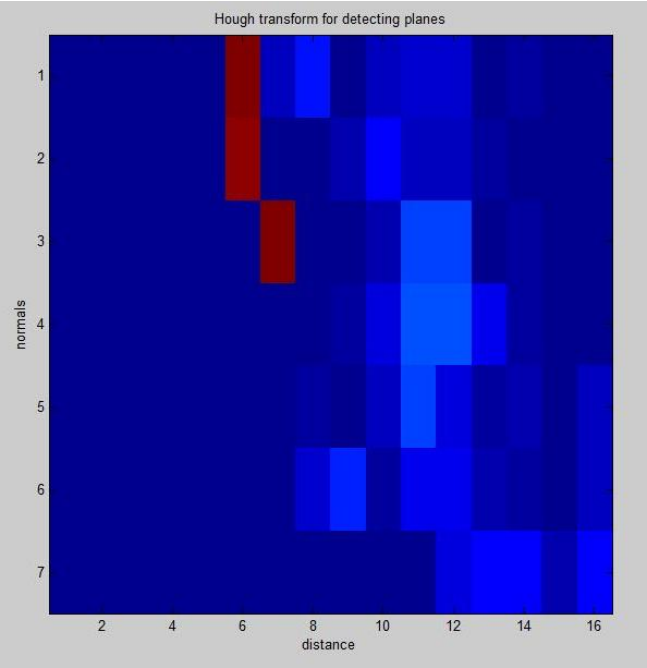


Fig. 5. Voting results from the Hough transform.

We have calculated the error by using the following formula:

error = normal * corresponding random points on that plane – distance from the origin.

We have set a threshold (0.01 in our case), if the error is within the threshold we consider the points to be lying on the corresponding plane. If not we ignore those points.

Undistort the planes using homography.

Once we detect the normals for each plane, we find the rotation and translation matrices between a virtual camera placed on the normal at a fixed distance from the plane.

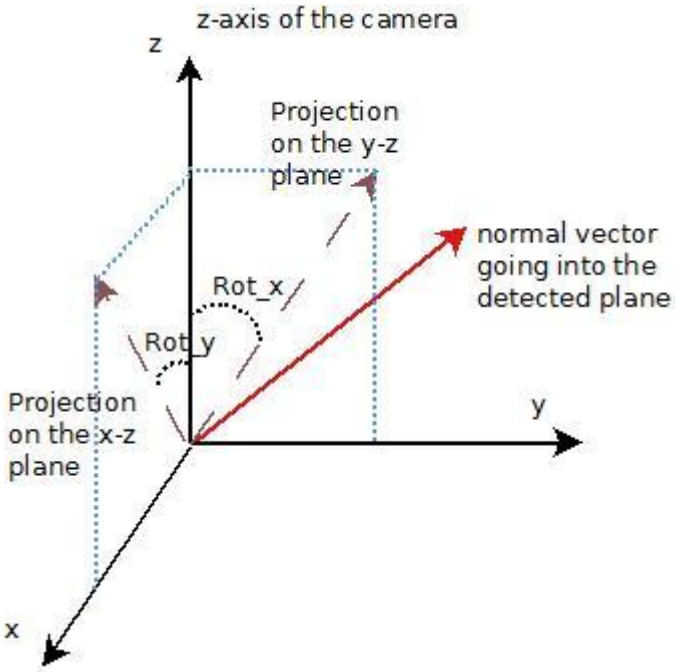


Fig. 6. Finding rotation between the pane and the camera.

We find the two rotations along x and y that will align the normal from the plane with the z-axis of the camera with the method depicted in fig. 6. The angles are determined with the dot product, and the signs by the cross product. The translation is easily determined as the position of the virtual camera is known to be at a fixed distance on the normal at the median of the detected plane.

Homography is used to relate any two images of the same planar surface. We need a virtual camera for estimating the homography. The virtual camera gives us an orthogonal view of the plane, which is essential in calculating the homography matrix. A homography matrix is estimated using the rotation and the translation vectors between the real camera and the virtual camera. We have used a RANSAC based algorithm by Peter Kovesi to find the homography.

The formulae related to these are:

$$\mathbf{H} = {}^{c'}_{c}\mathbf{R} - \frac{1}{d}{}^{c'}_{c}\mathbf{t}^{c}\mathbf{n}^{T}$$

$$ {}^{c'}\mathbf{x} \sim \mathbf{H}\,{}^{c}\mathbf{x} $$

We can then undistort the image from the Kinect (fig. 2) using this homography matrix. We have not performed this undistortion step. We have instead used EGT's perspective projection function to obtain the view from the virtual camera placed normal to the plane.

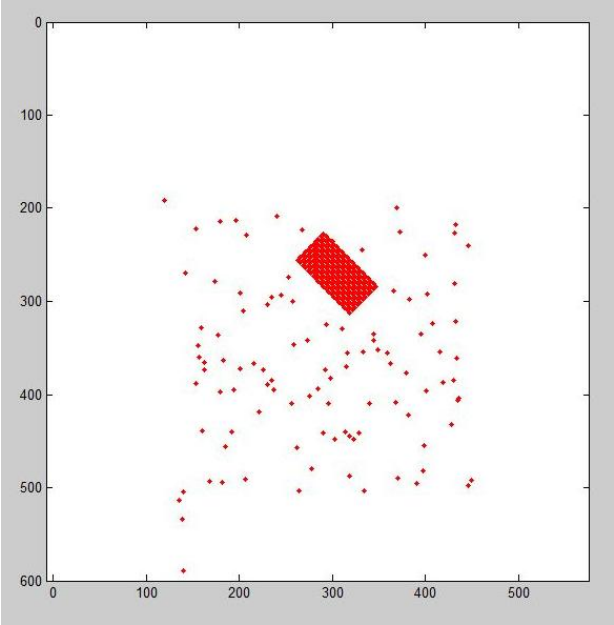
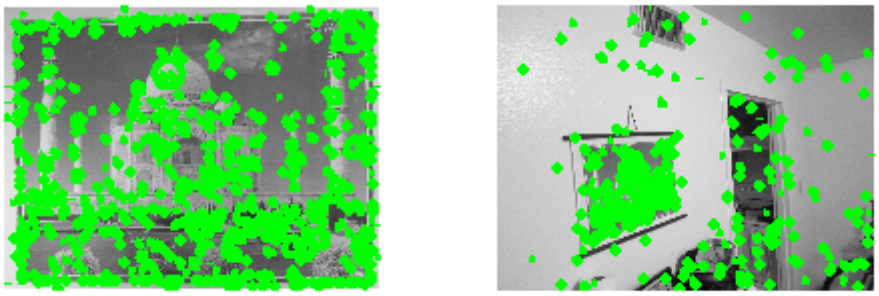


Fig. 6. Undistorted view of the bottom plane from perspective projection on the virtual camera. (This will actually be done using the homography matrix).

Extract SIFT features from the undistorted planes

We then use SIFT and RANSAC for matching our template with the undistorted scene. Since our simulations do not produce real images with texture that could be used for SIFT, it breaks our pipeline. We should be able to do this with Kinect. But we implemented SIFT based object recognition to keep the next stage ready for deployment. We have shown that it works nicely for matching objects in images with minor distortion. Figures 1 and 7 are examples of successful working of the SIFT matching code.



(a) (b)

(c)

Fig. 7. An example showing SIFT working even in a distorted image. (a) An image to be matched (b) Cluttered scene with the image, (c) Detected location of the image.

The SIFT matching worked in this case even though the wall was skewed, but the match was not perfect.

In future work, we plan to work with a real Kinect camera, and put the simulations to test.

References

- RANSAC by Peter Kovesi, http://www.csse.uwa.edu.au/
- http://ranger.uta.edu/~gianluca/teaching/CSE4392-5369_F12/
- VL_FEAT toolbox, http://www.vlfeat.org/

CODE ORGANIZATION

Requirements: EGT, VL_Feat, MATLABS's stats toolbox.

Folder 1 **– 3d_scene** - contains code to create the 3D scene and get the undistorted image:

1. generate_3d_room.m RUN ME! Does everything.
2. find_rotation.m – finds rotation between normals
3. connect_picture.m - helper
4. get_camera_axes.m - helper

Folder 2 – **sift_matching** - contains code for matching images using SIFT.

1. proj_sift_matching.m – RUN ME! Does the sift matching stuff.
2. appendimages.m. hnormalise.m , homography2d.m, iscolinear.m, normalise2dpts.m, ransac.m, ransacfithomography.m – RANSAC homography code by Peter Kovesi
3. poster.jpg, wall.jpg – sample images for matching