IMAGE RECONSTRUCTION FROM OMNI-DIRECTIONAL CAMERA

Kai Guo and Zhuang Li

Dec 15, 2007

Boston University

Department of Electrical and Computer Engineering

Technical report No. ECE-2007-06

BOSTON
UNIVERSITY

IMAGE RECONSTRUCTION FROM OMNI-DIRECTIONAL CAMERA

Kai Guo and Zhuang Li

Boston University
Department of Electrical and Computer Engineering
8 Saint Mary's Street
Boston, MA 02215
www.bu.edu/ece

Dec 15, 2007

Technical report No. ECE-2007-06

Summary

This project was completed as part of the assignment in EC 520. The objective of this project is to reconstruct the perspective image from the omni-directional image. We study the mapping relationship between the omni-directional image and panoramic image, and then we build the geometric projection relation between the panoramic image and the perspective image. The simulation results indicate that the method used for image reconstruction can efficiently correct the geometric distortion. In order to improve the perceptual image quality, we use cubic interpolator and image enhancement. Additionally, Circular Hough Transform is employed for automatic detection of the circle center.

Contents

List of figures

1 Introduction

With the widespread adoption of video sensor networks, it is desirable to maximum the covered area with minimum number of cameras, which can reduce the expense for cameras and the complexity to install and calibrate cameras. However, conventional cameras only have very limited field of view that leads to the increased network cost, need for mutual camera calibration and additional maintenance effort. Therefore, omni-directional (catadioptric) system is devised to solve this problem. A typical system uses a special mirror to reflect light rays from up to 360 degrees into a perspective camera lens and forms an omni-directional image (omni-image). This system is promising since it can be practically implemented in many areas, such as video conference, video surveillance, robotic vision and human activity monitoring. As long as the catadioptric imaging system is setup and calibrated, it can view the surroundings all the time. But the omni-images captured from catadioptric system have different properties compared with the perspective images in terms of lower effective resolution and imaging deformation. Such distortion leads to the difficulty for the images to be directly implemented; thus it is necessary to work out an efficient method to unwrap the omni-image. In this project, we firstly analyze the potential reasons that result in the image distortion, including geometric imaging deformation and sampling system discrepancy in the omni-image and panoramic image. Then we use a practical algorithm to unwrap the image by building the mapping relationship between two imaging systems. In order to further improve the unwrapping quality, several interpolation methods are compared and we can find that the cubic interpolation achieves the best performance. Although the panoramic image is much more comfortable for analysis than the omni-image, it is still not the best one for human observation; since it is not a perspective image. In order to solve this problem, we establish the relationship between the panoramic image and the perspective image which can be used for geometric correction. In practical unwrapping, an important prerequisite is to find the coordinates of the omni-image's center, which is always manually achieved. In this report, the circle

Hough transformation is used for auto center detection that has a satisfied accuracy.

2 Mirror design and constant vertical resolution

In this project the mirror of the catadioptric system has constant vertical resolution, which means that objects at a (pre-specified) fixed distance from the camera's optical axis will always be the same size in the image, independent of its vertical coordinates. The geometry of the image formation of the catadioptric camera is shown in Fig. 1. If we consider a cylinder of radius, *d*, aligned with the camera optical axis, we want to ensure that ratios of distances measured in the vertical direction along the surface of the cylinder would remain unchanged when measured in the image. Such invariance should be obtained by judiciously designing the mirror profile to yield a constant vertical resolution mirror. By using this kind of mirror, the imaging distortion can be avoided; therefore, we are able to develop appropriate formulas for unwrapping without introducing transformation error.

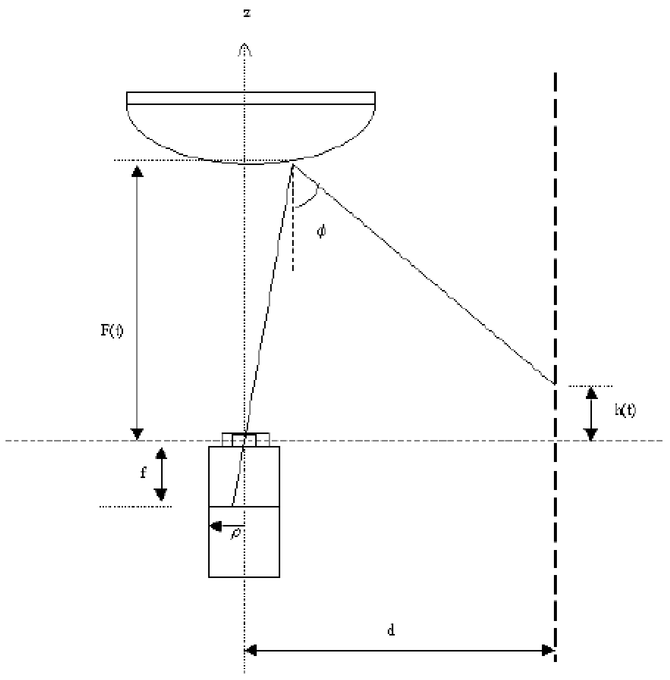

Fig. 1 Illustration of the constant vertical resolution

3. Panoramic image unwrapping method

The omni-image can be considered as the reflected cylinder (panoramic) image around the mirror, as shown in Fig. 2.

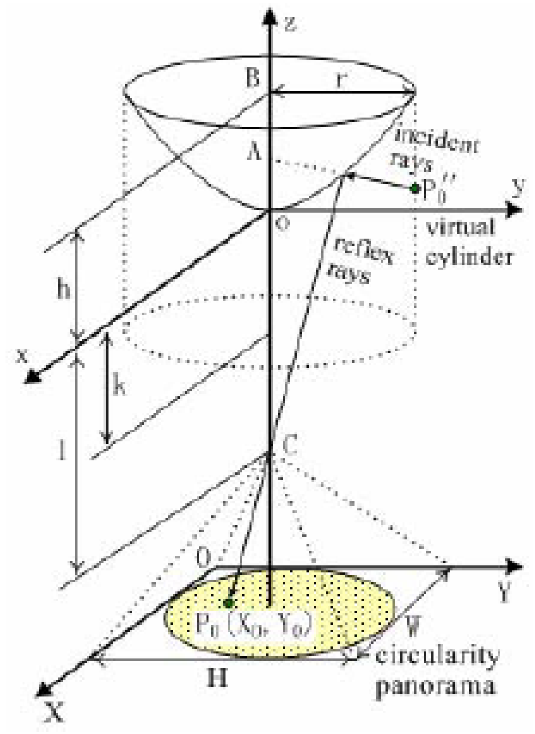

Fig. 2 The relationship between omni-image and the cylinder image

Fig. 2 implies that in each cutting surface, there exists a one-to-one mapping between the cylinder edge and the omni-image's radius, which can be used to recover the unwrapped image. Fig. 3 depicts the principle of panoramic unwrapping. Fig. 3 (a) denotes the omni-image with inner radius *r* and outer radius *R*. The region between *r* and *R* is the valid region that can be used for unwrapping, because the region inside the inner radius is usually the imaging of the camera lens that is not critical for unwrapping. Fig. 3 (b) shows the corresponding unrolled cylinder panorama. According to the fact discussed in section **II,** for a pixel $P_o^{'}(X_o^{'}, Y_o^{'})$ in the cylinder image, the polar coordinates of the corresponding pixel $P_o\,(X_o, Y_o)$ can be determined with the following formula:

$$h_o = \frac{r_m}{y_m} Y_o' \tag{1}$$

$$\theta_o = \frac{2\pi}{x_m} X_o' \tag{2}$$

Where (x_m, y_m) are the desired dimensions of cylinder image in pixel units, r_m is the radius of the mirror as seen in the circular panorama in pixel units, and θ_o is in radians. The correct ratio of x_m / y_m is 2π, because the out perimeter of the mirror corresponds to the bottom row of the unwrapped image.

Then we can find the mapping relationship between polar coordinates and rectangular coordinates in the circular panorama:

$$X_o = X_c + h_o \cos\theta_o = X_c + \frac{r_m}{y_m} Y_o' \cos\frac{2\pi}{x_m} X_o' \tag{3}$$

$$Y_o = Y_c + h_o \sin\theta_o = Y_c + \frac{r_m}{y_m} Y_o' \sin\frac{2\pi}{x_m} X_o' \tag{4}$$

According to the formula above, it is easy to calculate the coordinate in the omni-image when given a point in the cylinder panoramic image. Note that (X_o, Y_o) is usually not an integer pixel; therefore, the pixel values cannot be directly obtained from the image captured by the camera. Therefore, we need to use some interpolation approaches to estimate the pixel values.

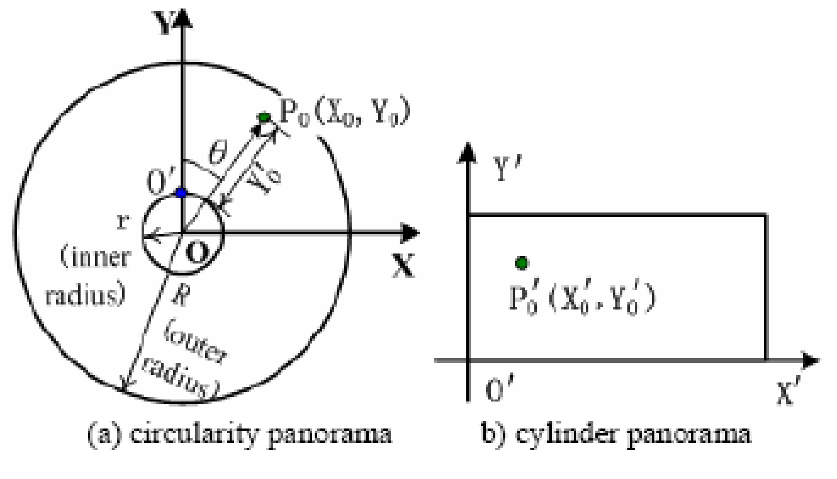

Fig. 3 Omni-image unwrapping

4 Analysis of panoramic image unwrapping error

In order to correctly unroll the omni-image, it is worthwhile to analyze the possible reasons that result in the panoramic image unwrapping error. Based on the discussion in Section II, we have known that the coordinates in the omni-image mapped from cylinder panorama are fractional values that can result in the unwrapping error. If a polar sensor with sufficient high resolution is used to capture the omni-image, this sort of error can be efficiently restricted. However, most of the commercial cameras are based on the orthogonal sampling technique; therefore, the captured omni-image has lower effective resolution near the mirror vertex than near the mirror bottom, which is one of the major sources of the unwrapping error.

The other major source of error can be analyzed according to the geometric imaging principle. This sort of error depends on the shape and position of the mirror and the influence of both factors will be discussed below.

It is well known that the virtual image of a plane mirror has no deformation, i.e., no stretching or shrinking in the size and no shape distortion. On the contrary, a curved mirror will always lead to the virtual imaging distortion. The conic mirror can be considered as a combination of two plane mirrors in the cutting surface; thus, the conic mirror has no imaging distortion, while other curved mirrors (such as hyperbolic mirror, parabolic mirror and hemisphere mirror) have such sort of distortion.

Despite the conic mirror can avoid the virtual imaging distortion, there is still another possibility can lead to the panorama distortion, as illustrated in Fig. 3. In Fig. 3 an object AC is projected onto the conic mirror and reflected to form an omni-image on the horizontal camera sensor and B is the center of the object. We can find that B is still the center point in the virtual image which indicates that conic mirror has no imaging distortion. Nevertheless, when the virtual image is projected onto the sensor, B is no longer the center point of AC, because the virtual image is not parallel to the horizontal sensor. If the sensor is rotated from AC to A'C, there is no projection error any more and B' is the center point of A'C.

We have analyzed the possible reasons to lead to the omni-image distortion and in the next section we will discuss the efficient solution to reduce the distortion.

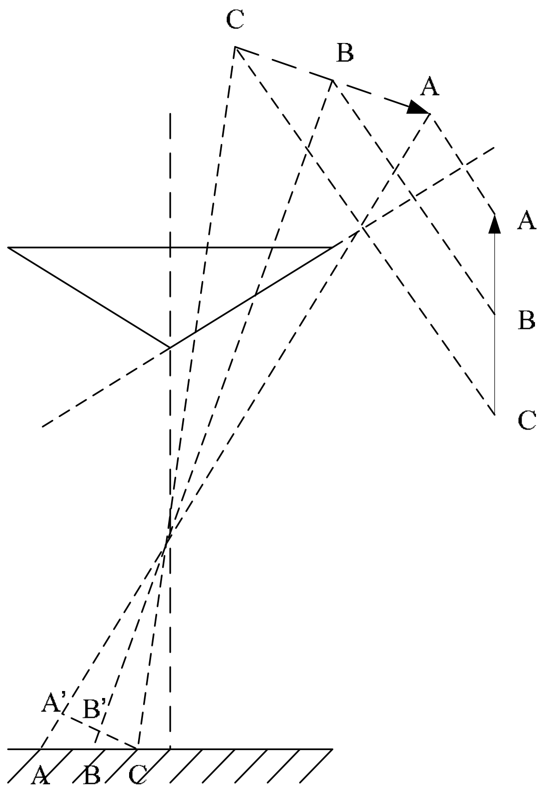

Fig. 4 Illustration of panorama distortion

5 Interpolation methods

Although constant vertical resolution mirror can avoid imaging distortion, the calculated pixels in the circular image does not correspond exactly “one to one” to the pixels of panoramic image so sub pixel anti-aliasing methods should be used. Interpolation is such a method which is critical for approaching the better image quality from the unwrapped image. The ideal interpolation is the upsampling technique, but in spatial domain the *SINC* function is required for upsampling which makes use of all the pixels to estimate an arbitrary pixel value, leading to the overload of computational complexity. For practical implementation, several interpolation methods are used to approximate the current pixel value from its neighborhood pixels.

1) Nearest-neighbor interpolation

The nearest neighbor algorithm simply selects the value of the nearest point, without considering the values of other neighboring points at all. (e.g. if *x*=2.4 lies between 2 and 3, then *x* is equal to2).

2) Average (mean) interpolation

The algorithm simply calculates the mean value of the neighboring points of the given point. The average (mean) filter can smooth image data and reduce the noise.

3) Linear interpolation

Linear interpolation is the simplest method of getting values at positions between the data points. The points are simply joined by straight line segments. In demanding situations, linear interpolation is often not accurate enough.

4) Bilinear interpolation

Bilinear interpolation is an extension of linear interpolation for interpolating functions of two variables on a regular grid. The key idea is to perform linear interpolation first in one direction, and then in the other direction. The result of bilinear interpolation is independent of the order of interpolation.

Fig.6 shows that the four red dots show the data points and the green dot is the point at which we want to interpolate. Bilinear interpolation first interpolates horizontally between the red points (Q_{11} Q_{21} and Q_{12} Q_{22}) to determine the blue points R_1 andR_2, respectively. Then it interpolates vertically between the blue points to determine the green point.

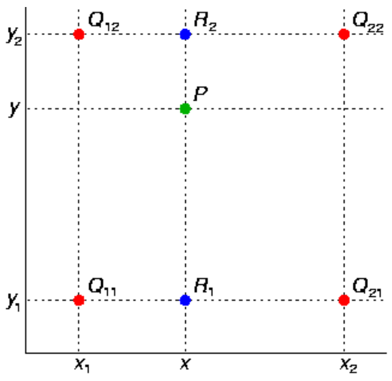

Fig.5 Bilinear interpolation

5) Cubic interpolation

Cubic interpolation is the method that offers true continuity between the segments. It requires more than just the two endpoints of the segment and the two points on either side of them. The points are joined by 3rd order polynomial segments.

Fig.6 shows that interpolation on a data set (red points) consists of pieces of polynomials (blue lines). Any point (purple point) can be calculated by the polynomials. The red line and blue line represent linear interpolation and cubic interpolation, respectively. The cubic interpolation is deemed as a good approximation of *SINC* function.

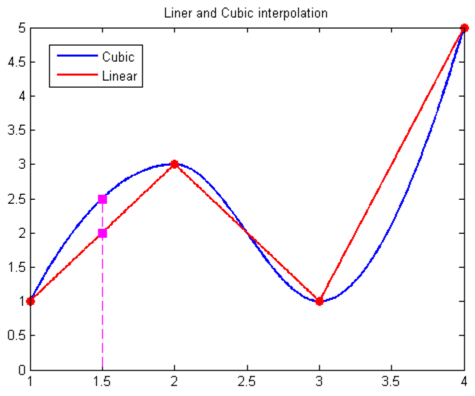

Fig.6 Linear and Cubic interpolation

The 2-D cubic interpolation interpolates between data points. It finds values of a two-dimensional function underlying f(x, y) the data at intermediate points.

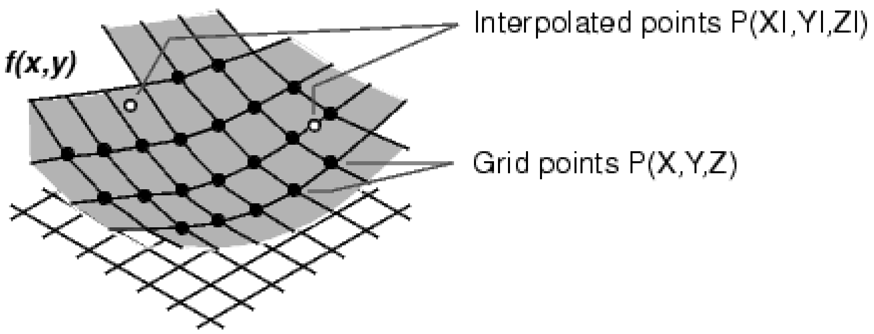

Fig.7 2-D Cubic interpolation

6 Perspective projection from panoramic image

In the previous sections, the omni-image has been unwrapped to panoramic image that is reconstructed onto the cylinder around the catadioptric system. However, the panorama may lead to some annoyed geometric distortion and is not appropriate for human observation; thus, it is desirable to form the perspective image by projecting the cylinder image onto a virtual image plane.

Limited by the nature of the perspective image, the reconstructed image is usually confined to a narrow range. The 3-D mapping relationship between panorama and perspective image is shown in Fig. 8, and the 2-D planform is shown in Fig. 9. In Fig. 8, the plane OXY is the perspective imaging plane; MNST is the curved plane of the cylinder which is used to reconstruct the perspective image. PQ is the axial line of the cylinder. V is the virtual view point. Since the mirror used in our report (constant vertical resolution) has no focus point, our catadioptric camera is not a single view system. Therefore, it is unlikely to form a perfect perspective image and what we can do is to minimize the imaging error by adjusting the position of the virtual view point V, which is always a manual job. VAA' and VBB' are the projection lines, where points A, B are on the cylinder, and points A', B' are on the plane. According to the geometry projection principles, we can formulate the relationship between the coordinate $A'(x_p, y_p)$ and the coordinate $A(x_c, y_c)$.

$$\alpha = \frac{L}{d_c} \tag{5}$$

$$d_p = d_c + s \tag{6}$$

$$width = 2d_p \tan\frac{\alpha}{2} \tag{7}$$

$$d = \frac{d_p}{\cos(\alpha/2 - x_c/d_c)} \tag{8}$$

$$x_p = \frac{width}{2} - d_p \tan(\frac{\alpha}{2} - \frac{x_c}{d_c}) \tag{9}$$

$$y_p = y_v + \frac{d}{d_c}(y_c - y_v) \tag{10}$$

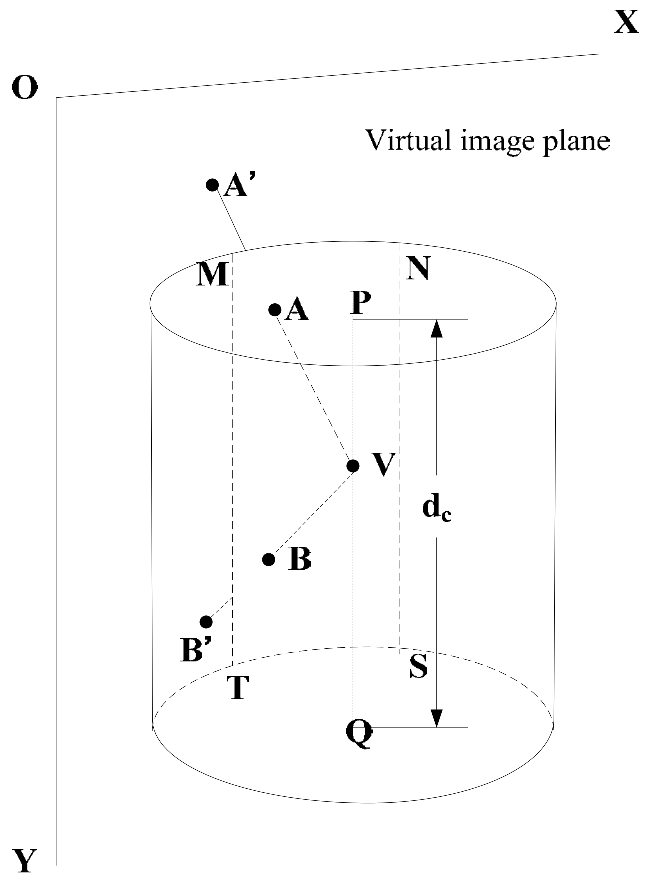

Fig. 8 Projection from the panorama to the virtual plane

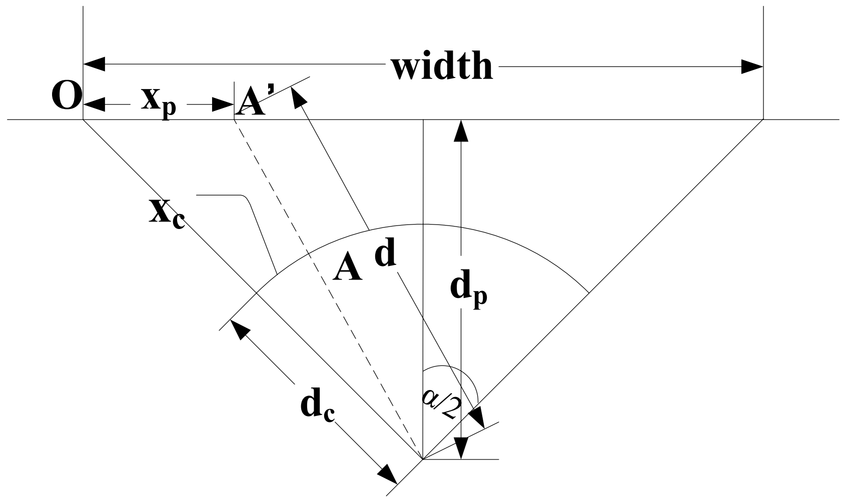

Fig. 9 Planform of the projection from the panorama to the virtual plane

where d_c is the radius of the cylinder, d_p is the perpendicular distance from the circle center to the perspective plane, y_v is the coordinate of the virtual view point which shares the same *x* coordinates with the circle center. Reformulate (9) and (10):

$$x_c = d_c[\frac{\alpha}{2} - \frac{1}{d_p}\tan^{-1}(\frac{width}{2} - x_p)] \tag{11}$$

$$y_c = y_v + \frac{d_c}{d_p}(y_p - y_v) \tag{12}$$

Given a point on the perspective plane, we can calculate its corresponding mapping point on the panorama, whose coordinates are usually fractional values. Thus, it is necessary to estimate the pixel value of the fractional coordinates with interpolation technique.

7 Color Enhancement

Image enhancement is a powerful tool for the improvement of digital image quality (for visual inspection or for machine analysis), without knowledge about the source of degradation. In this report, one of the image enhancement algorithms, color-mapping transformation, is implemented.

As is known that the contrast-stretching technique can be used to increase the dynamic range of gray-levels, it is natural to extend it to the color image processing. The color image transform is in the form: $s_i = T_i(r_i),\ i = 1, 2, 3, \ldots, n$ where r_i and s_i are the color components of the input and output images, *n* is the dimension of the color space of r_i, and T_i is referred to as full-color transformation (or mapping) functions.

Here cubic-spline interpolation mapping function is used to adjust the contrast and luminance of images, so that the histogram is smoothed.

Fig. 10 shows the s-shaped mapping function.

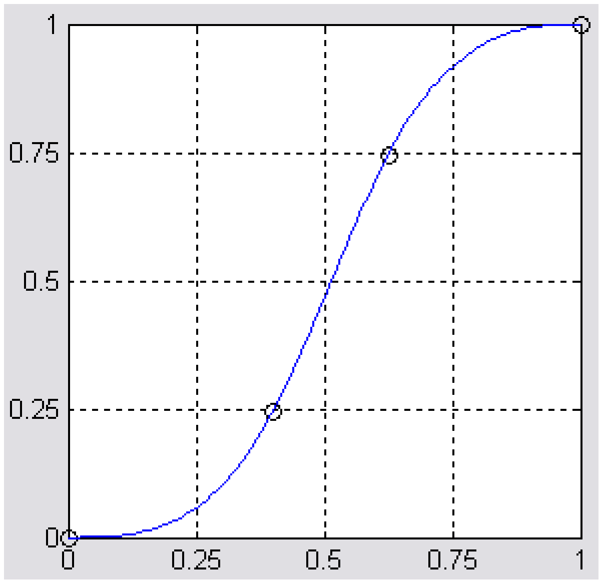

Fig. 10 S-shaped mapping function

This mapping function maps the input image with a full-color reference bar. The two ending points can be used to adjust the luminance by moving up and down; while the two middle points can be used to adjust the image contrast. The adjusting work is manually processed until a satisfied image is obtained and the evaluation of image quality is a highly subjective process.

8 Automatic circle center detection

The circle center position and the radius of the circular image are critical for image unwrapping, and in some circumstances the manual search of circle center may not be feasible, thus it's worthwhile to develop an algorithm that can automatically detect those parameters in an efficient and accurate way. One of the prevalent tools to achieve this goal is the Circle Hough Transform (CHT), which can accurately find the circle centers and radii. Besides, the Hough technique has been found to be effective and highly robust against occlusions, noise and other artifacts. The CHT is illustrated in Fig. 11. A set of edge points within the original image, are indicated by the black circles. Each edge point in the original image contributes a circle of radius *R* to the output accumulator space, and this contribution is shown by the gray circles. The output accumulator space has a peak where these plotted circles overlap. In the other word, the peak is just the center of the circle.

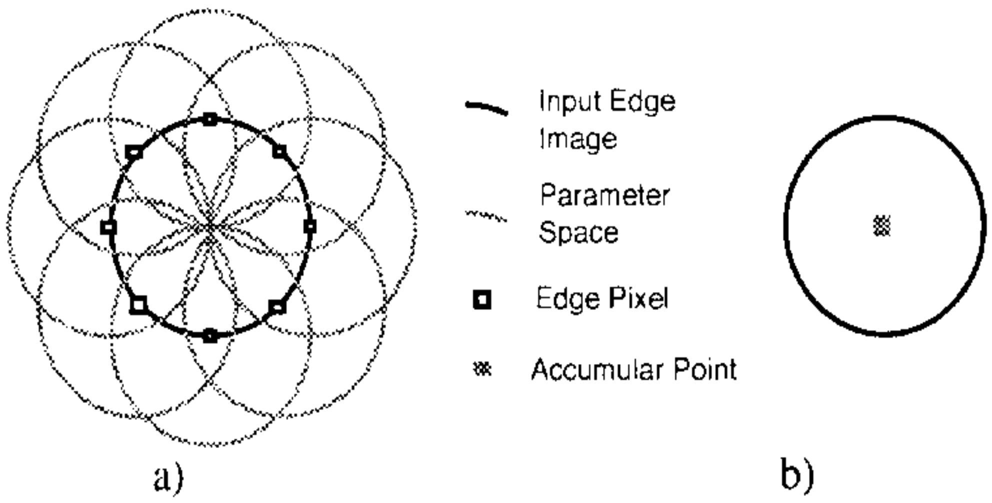

Fig. 11 Illustration of Circle Hough Transform

For the Circle Hough Transform calculation, a separate circle filter can be used for each radius of circle to be detected. This can be used to form the 3-dimensional parameter space - a_x, a_y and *R* ($a_x - a_y - R$ domain), where two dimensions represent the coordinates of the circle center (a_x,a_y), and the third stands for its radius (*R*):

$$(x - a_x)^2 + (y - a_y)^2 = R^2 \tag{13}$$

In (13), (a_x, a_y) and *R* are the parameters and *x, y* are the variables correspond to the points of the circle in the *X*-*Y* domain. For implementing Circle Hough Transform, we need to re-analyze (13) in the $a_x - a_y - R$ domain, where a_x, a_y, R are regarded as variables and *x, y* are parameters. In this new domain, we need to find the peak pixel value, which is mostly likely to be the center of the circle. The center-detection algorithm is based on the edge image (a binary image), which can efficiently reduce the computational complexity and improve the detection accuracy.

9 Simulation results

In our experiment, we carry out the unwrapping from the omni-image to the panoramic image, and then transform the panoramic image to the perspective image. Different interpolation methods (nearest-neighbor estimator, average filter, linear interpolator and cubic interpolator) are compared. Fig. 12 shows the original omni-images and the unwrapped panoramic images are shown in Fig. 13- Fig. 16, corresponding to different interpolation methods. Due to the limitation of the page size, the panoramic image is only displayed at 20%; therefore, the difference is not visually distinct. With careful comparison of the full size images, we can find that the cubic interpolation leads to the best visual quality. Thus, the following simulations are all based on the cubic interpolator. Fig. 17 shows the images after doing the image enhancement, which can efficiently improve the visual quality. Fig. 18 shows the perspective images with different positions of the view points. It is obvious that the view point is significant to the reconstructed image. In this example, when the view point is very high (as shown in Fig. 18 (a)), the perspective image is concave distorted and vice versa. When the view point is properly selected ($y_v = 0.65d_c$), the distortion is minimized; because the wall lines and the fluorescent lamps are straight, which is consistent with the property of the perspective image. Then we carry out the auto-center detection and the experimental results are shown in Fig. 19, in which the centers are marked by green cross. The results indicate that the CHT algorithm has a good approximation of the circle center.

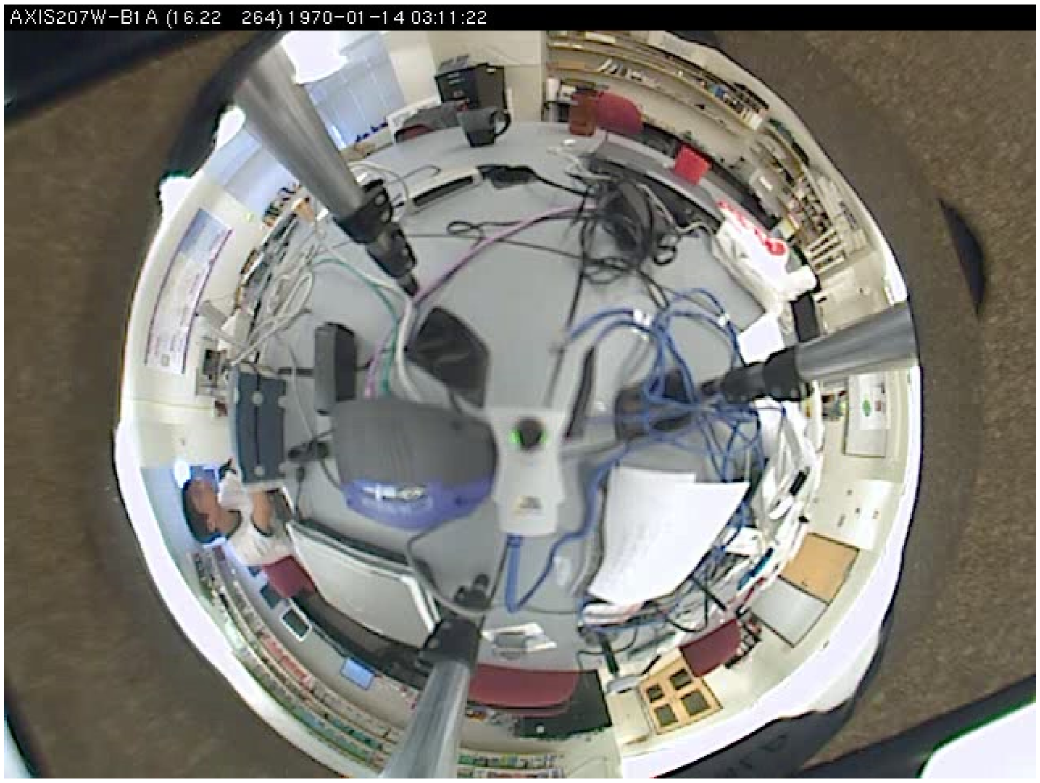

Fig. 12 (a) Omni-image: Lab

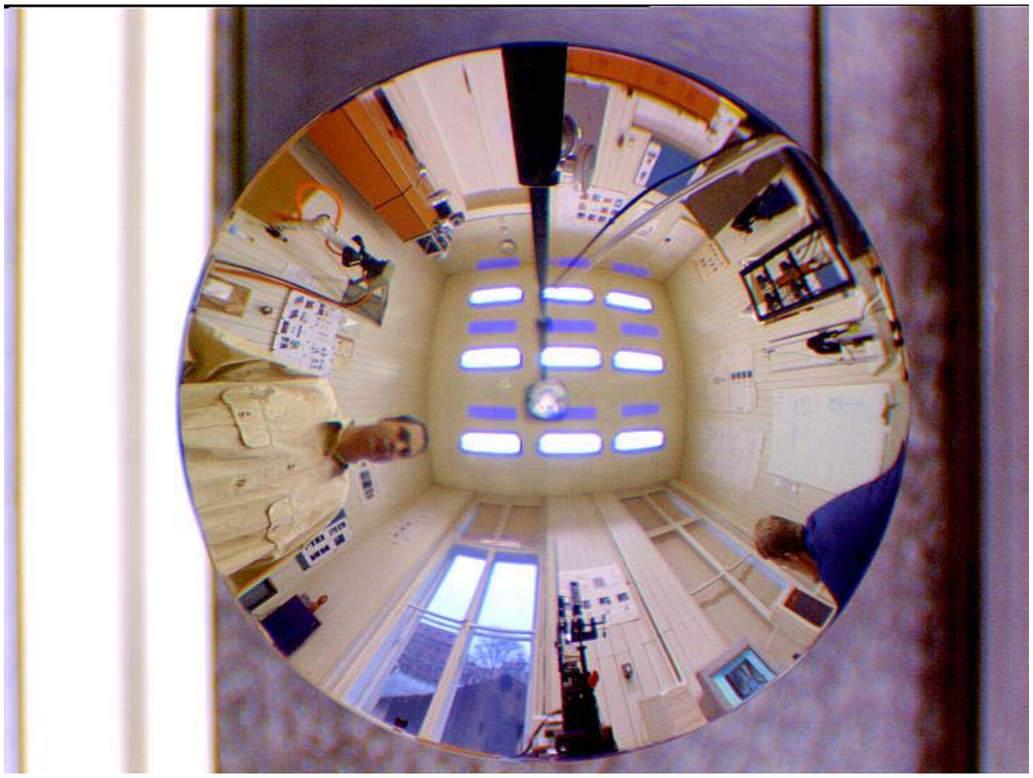

Fig. 12 (b) Omni-image: Office

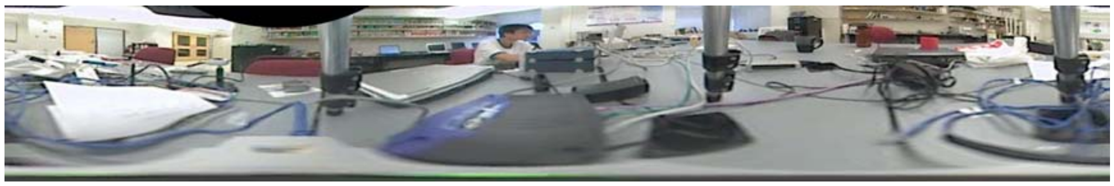
Fig. 13 (a) Panoramic image: Lab (nearest-neighbor method)

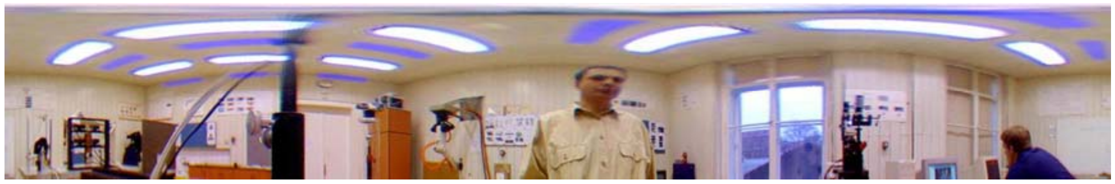
Fig. 13 (b) Panoramic image: Office (nearest-neighbor method)

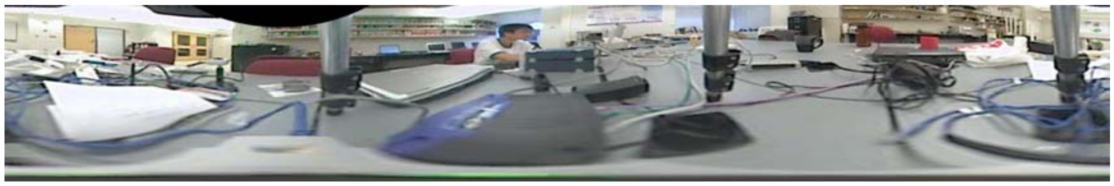
Fig. 14 (a) Panoramic image: Lab (average filter)

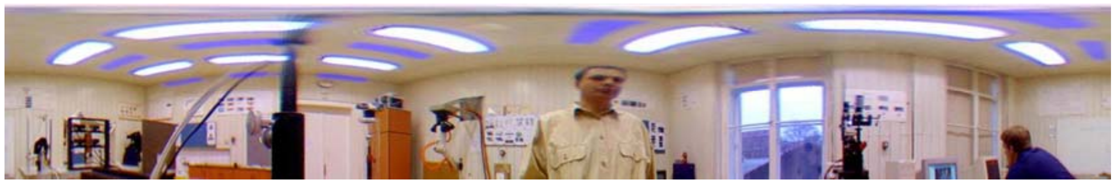
Fig. 14 (b) Panoramic image: Office (average)

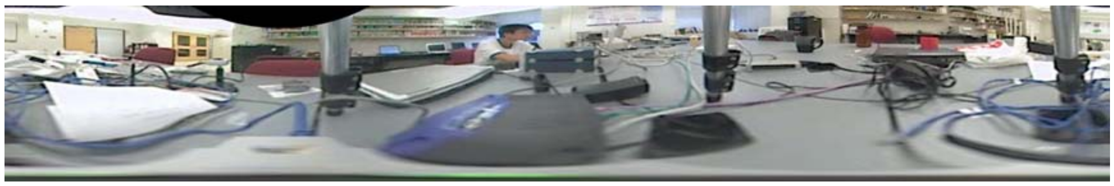
Fig. 15 (a) Panoramic image: Lab (linear interpolator)

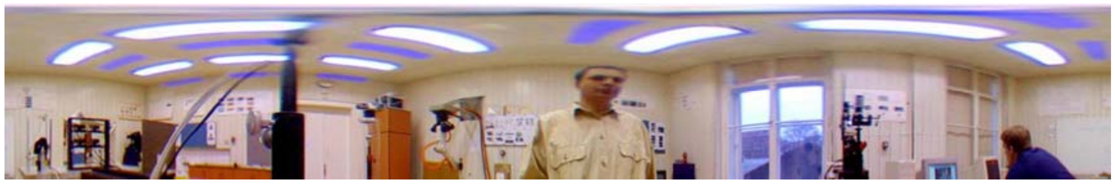
Fig. 15 (b) Panoramic image: Office (linear interpolator)

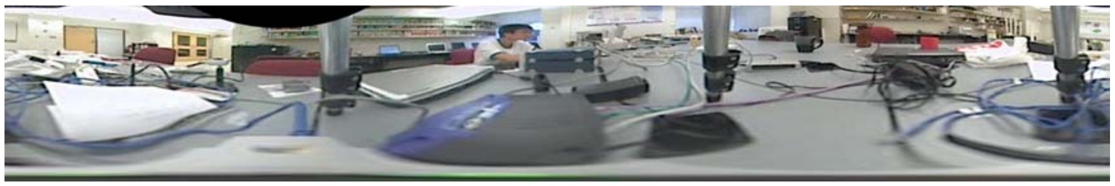
Fig. 16 (a) Panoramic image: Lab (cubic interpolator)

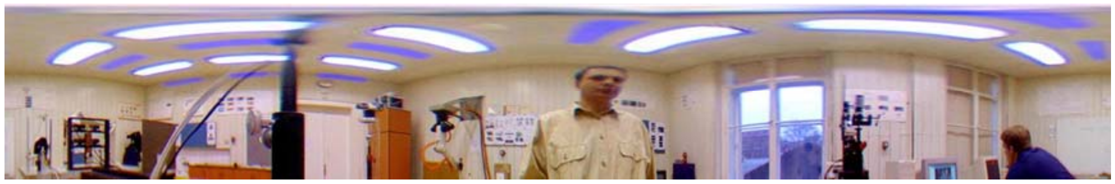
Fig. 16 (b) Panoramic image: Office (cubic interpolator)

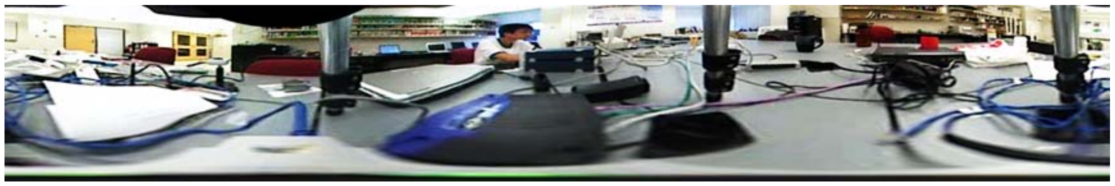
Fig. 17 (a) Panoramic image: Lab (cubic interpolator + image enhancement)

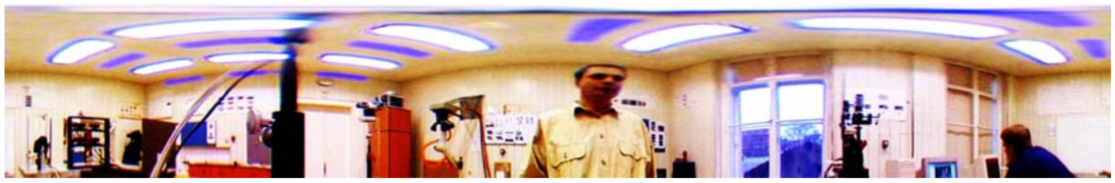
Fig. 17 (b) Panoramic image: Office (cubic interpolator + image enhancement)

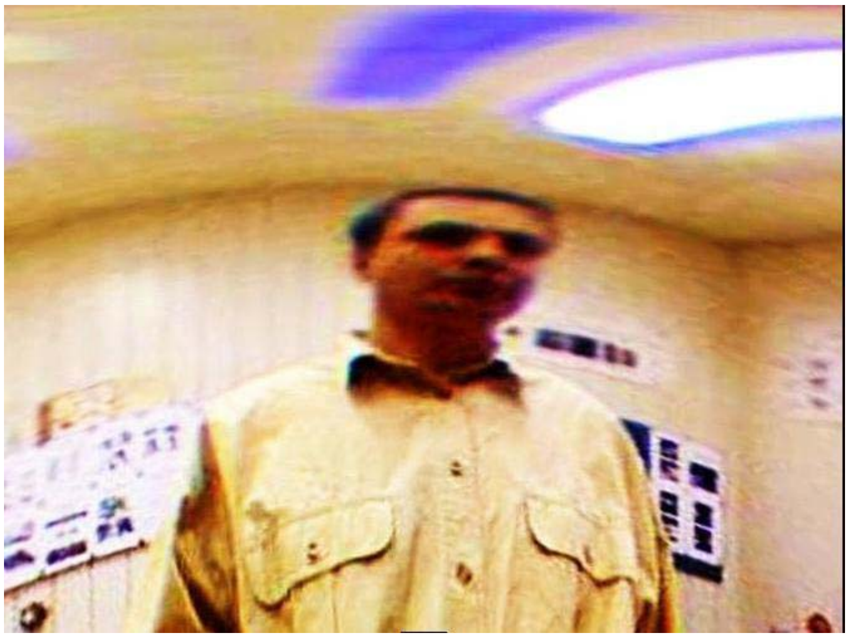

Fig. 18 (a) Perspective image: Lab (view point position: $y_v = 0.10 d_c$)

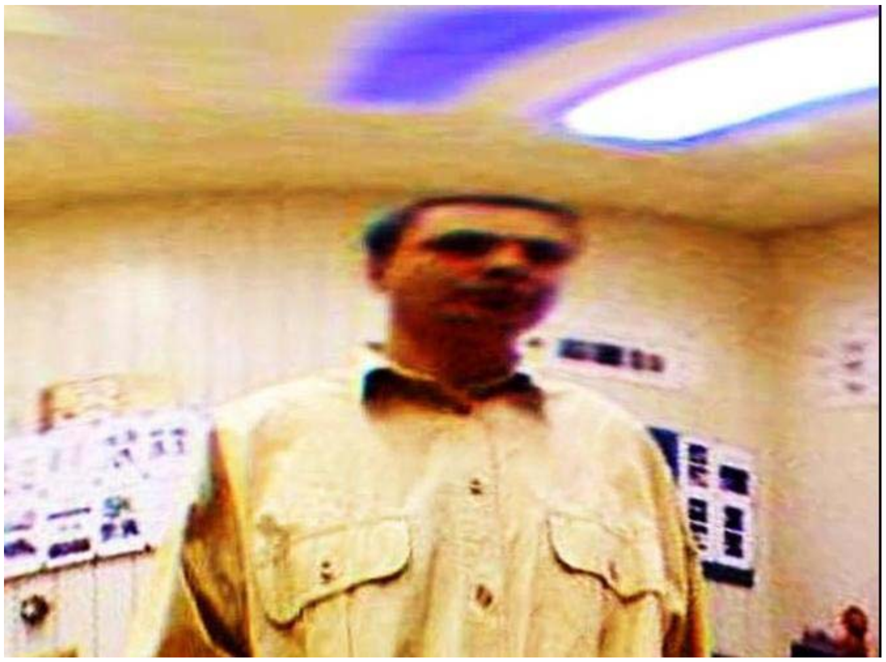

Fig. 18 (b) Perspective image: Lab (view point position: $y_v = 0.30 d_c$)

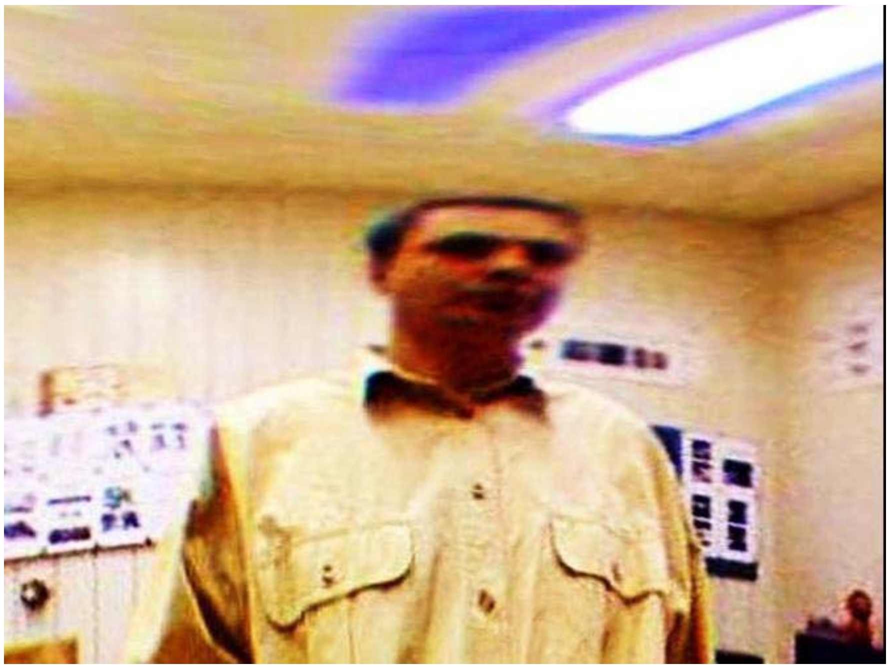

Fig. 18 (c) Perspective image: Lab (view point position: $y_v = 0.50 d_c$)

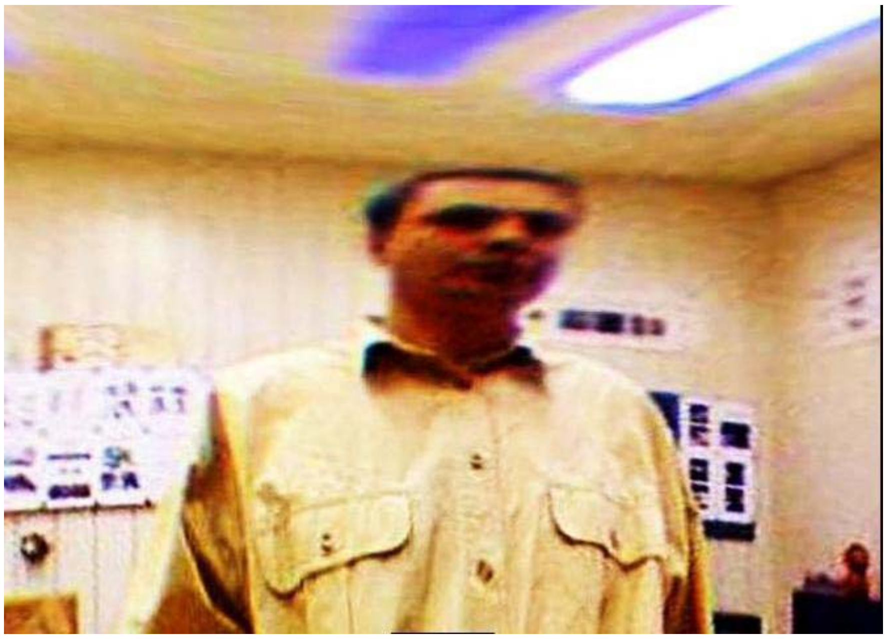

Fig. 18 (d) Perspective image: Lab (view point position: $y_v = 0.65 d_c$)

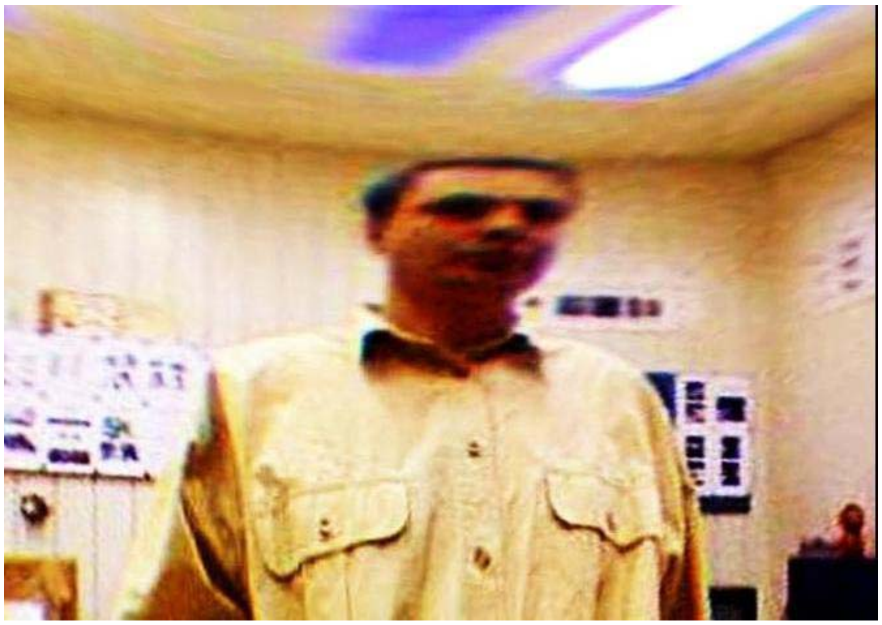

Fig. 18 (e) Perspective image: Lab (view point position: $y_v = 0.65d_c$)

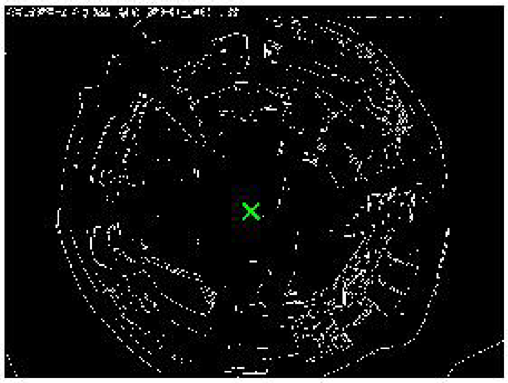

Fig 19(a) Auto-detected circle center: Lab

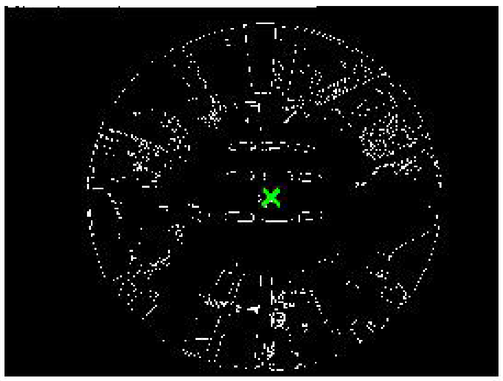

Fig 19(b) Auto-detected circle center: Office

10 Conclusion

The work in this report deals with the set of methods for omni-directional image processing. Firstly, we analyze the reasons that result in the imaging distortion of the catadioptric system. Then we establish the mapping relationship between the omni-image and the panoramic image under the circumstance of the constant vertical resolution mirror. We also build the geometric projection relation between the panoramic image and the perspective image within a narrow range. The simulation results demonstrate that the image unwrapping can efficiently correct the geometric distortion. In order to improve the image quality, different interpolation algorithms are implemented and we find that the cubic interpolator outperforms others. Furthermore, the image enhancement is used to improve the subjective visual effect by adjusting the contrast and luminance of the image. Finally, we use Circular Hough Transform to automatically detect the center of the circle. The simulation results imply that the CHT has fairly good performance.

Appendix

Below is listed Matlab source code developed for this project.

I. Omni-image -> panoramic image

```
%omni-directional image processing
%This fuction is used to unwrap the circular image to its panoramic image.
%The proportion of the output panoramic image can be defined by user.

clc
clear
circle=double(imread('color.jpg'));

%inner radius: r1
%outer radius: r2
%coordinates in circle: Xc, Yc
%coordinates in square: Xs, Ys
%angle in circle: theta
%center point in circle: centerX, centerY
%length of panoramic image: Length
%width of panoramic image: Width
%length and width adjuster: l_times, w_times

r1=10;
r2=272;
centerX=272;
centerY=272;
l_times=1;
Length=round(l_times*2*pi*r2);
w_times=1;
Width=round(w_times*(r2-r1));
```

```
for w=1:Width
    for l=1:Length

        Ys=r2-w/w_times;
        theta=2*pi*l/Length;
        Xc=centerX+Ys*cos(theta);
        Yc=centerY+Ys*sin(theta);
        xi(w,l)=Xc;
        yi(w,l)=Yc;

    end
end

square(:,:,1)=interp2(circle(:,:,1),yi,xi,'cubic');
square(:,:,2)=interp2(circle(:,:,2),yi,xi,'cubic');
square(:,:,3)=interp2(circle(:,:,3),yi,xi,'cubic');
square=uint8(square);

figure,imshow(square,[])
```

II. Panoramic image -> Perspective image

```
%omni-directional image processing
%This fuction is used to generate perspective image from its panoramic image.
%User needs to define a paortion of panoramic image which will be used for
%generating the perspective image.

%user defined portion of the panoramic image: pano
%The height and radius of panoramic cylinder: dc
%view point position: yv
%view point position adjuster: percent
%width of the perspective image: width
```

```
%height of the perpective image: height
%maximum length of the perpective image: vertical_max
%height of the perpective image adjuster: vertical_offset

clc
clear
close all

square=imread('final1_4_1.jpg');
dc=size(square,1);
pano=double(square(1:300,1455:2023,:));
[pano_W,pano_L,h]=size(pano);

alpha=pano_L/dc;
dp=dc+30;
percent=0.6;
yv=dc*percent;
width=floor(2*dp*tan(alpha/2));
vertical_offset=0;
vertical_max=round(yv+dp/dc*(dc-yv));
height=vertical_max-vertical_offset;

for xp=1:width
    for yp=1:height

        xc=dc*(alpha/2-atan((width/2-xp)/dp));
        d=sqrt((width/2-xp)^2+dp^2);
        yc=yv+(yp-yv)*dc/d;
        xi(xp,yp)=xc;
        yi(xp,yp)=yc;
    end
end

pesp(:,:,1)=(interp2(pano(:,:,1),xi,yi,'cubic'))';
pesp(:,:,2)=(interp2(pano(:,:,2),xi,yi,'cubic'))';
pesp(:,:,3)=(interp2(pano(:,:,3),xi,yi,'cubic'))';

pesp=uint8(pesp);
imshow(pesp(1:pano_W,:,:),[])
```

III. Circle center detection

```
% function result =
hough_circle(I,step_x,step_y,step_r,step_angle,r_min,r_max,p)
clear
clc

aa=imread('color2.jpg');
a=aa(:,:,1);
b=edge(a,'sobel');

I=b;
step_x=1;
step_y=1;
step_r=1;
step_angle=0.05;
[h,w] = size(I);
r_min=max(h,w)/4;
r_max=max(h,w)/2;

p=1;
```

```
size_x = round(h/2/step_x)+1;
size_y = round(w/2/step_y)+1;
size_r = round((r_max-r_min)/step_r)+1;
size_angle = round(2*pi/step_angle);
hough_space = zeros(size_x,size_y,size_r);
[rows,cols] = find(I);
ecount = size(rows);

hh= waitbar(0,'Please wait...');

for i=1:ecount
    for j=1:size_r
        for k=1:size_angle
            x = round(rows(i)-(r_min+(j-1)*step_r)*cos(k*step_angle));
            y = round(cols(i)-(r_min+(j-1)*step_r)*sin(k*step_angle));
            if(x>h/2-h/4&x<h/2+h/4&y>w/2-w/4&y<w/2+w/4)
            hough_space(x-round(h/4)+1,y-round(w/4)+1,j)=
            hough_space(x-round(h/4)+1,y-round(w/4)+1,j)+1;
            end
        end
    end
     waitbar(i/100)
end
max_para = max(max(max(hough_space)));
%store the index for potential peaks in parameter space
index = find(hough_space==max_para);
length = size(index);

for k=1:length
    par3 = floor(index(k)/(size_x*size_y))+1;
    par2 = floor((index(k)-(par3-1)*(size_x*size_y))/size_x)+1;
    par1 = index(k)-(par3-1)*(size_x*size_y)-(par2-1)*size_x;
    par11=ceil(par1+h/4);
    par22=ceil(par2+w/4);
    fprintf(1,'Center %d %d radius %d\n',par11,par22,r_min+(par3-1)*step_r);
end

close (hh)
```