Tracking a falling object by stereo camera

Jiaxin Xu, Runnan Zou
Department of Mechanical Engineering, University of Ottawa
Ottawa, Canada
jxu171@uottawa.ca,
rzou043@uottawa.ca

***Abstract*—With rapid development and worldwide application of autonomous driving and robotics, machine vision is becoming an essential method for object detection. In this project, a basketball is tracked in its falling way by method of machine vision. A stereo camera tracking system is designed and applied in tracking falling of the basketball. Stereo camera calibration, image recognition and 3D reconstruction are utilized in this project. With the establishment of a stereo camera system, the trajectory of ball-falling is detected. The information of the falling ball is derived in aspect of position, attitude, velocity and acceleration. Two experiment are conducted to validate the effectiveness of the system. Results show that the designed system can acquire a relatively high accuracy result. The obtained trajectory and attitude are consistent with observation. The velocity and acceleration of the ball is in accordance with the law of free fall.**

Keywords—Stereo Camera, Calibration, 3D Reconstruction, Free Fall

I. INTRODUCTION

Tracking a moving object is an important research topic in the computer vision area. It is able to be applied in different areas. In the field of environment, computer vision is widely used in remote sensing analysis, meteorological monitoring, and other aspects. Great contributions has been made to natural disaster prediction, resources, and ecological environment detection. In the field of production and manufacturing, it is equipped with computer vision operations. In the field of national defense, computer vision plays an irreplaceable role, especially in various applications related to images, such as missile guidance and target tracking. As early as 1960, people began to carried out preliminary study of computer vision. In the early stage of its development, it mainly focused on the study of two-dimensional images. In 1982, Marr put forward his computer vision theory, which is the most complete computer vision theory by summarizing all the important achievements in neuroscience, psychology, and other fields. In Marr's theory of computer vision, the visual process is usually divided into three stages [1]. The first stage is the processing of the original image, including edge detection, feature point extraction, etc. The second stage is to recover the target information including the contour and the depth of the visible part through the original image processed in the first stage from the observer's point of view, but these are not real three-dimensional representation, so it is called 2.5D, which is the last stage is to recover the three-dimensional information of the real object based on the above two stages. Although Marr's computer vision theory has great significance, it still has limitations of the times. After the 1990s, researchers gradually put forward a series of more practical theories and methods, among which the more typical is the active vision theory. The main feature of active vision is that it does not need to reconstruct the environment, divide the visual tasks according to specific purposes and carry out the independent perception of different tasks. Through the study of these theories, researchers have made great progress in the field of computer vision.

Three-dimensional reconstruction is the inverse process of the imaging process. The process of recovering 3D information from 2D projection is called 3D reconstruction which increases the accuracy of tracking.. In the field of computer vision, 3D reconstruction is an important research topic which is often applied in visual navigation, virtual enhancement, slam, assisted surgery, and other fields. In application of 3D reconstruction, the geometric characteristics, surface properties, and camera parameters of the target object are inversely deduced from the two-dimensional image [2]. The solution process of 3D reconstruction is nonlinear, so the solution is often not unique, and it is extremely sensitive to the existence of noise. Therefore, an important advanced technology that integrates the development of optics, electronics, and computer technology is to recover the three-dimensional information of objects in real space through two-dimensional image data, which has attracted extensive attention in the world. In 1992, Hartley and Faugras first proposed the theory of 3D reconstruction based on an image sequence [3] [4]. In Hartley and Faugras' paper, 3D reconstruction is summarized as three steps: the estimation of the basic matrix, the calibration of camera parameters, and the calculation of the projection matrix and corresponding three-dimensional space coordinates.

This project aims to detect the position and attitude of a falling basketball and calculate the velocity and acceleration based on a stereo camera system with the designed algorithm. Based on requirements for this experiment, an algorithm is proposed and it is shown as flow chart in Fig. 1.

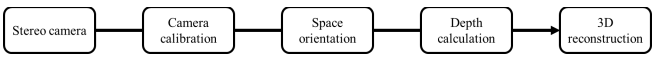

Fig. 1. Algorithm for this project

The main applicated process should be divided into three main steps: camera calibration, tracking, and 3d reconstruction.

Because of the methods chosen for the result analyzing, we designed our experiment for three main purposes. The first one is to make the whole videos can record the completed falling process as much as mobile phones' cameras can. Secondly, the basketball tracked here should easy to be identified, in other words, the background of the environment can keep simple. Moreover, due to the stereo camera system, the two mobile phones' camera should take videos simultaneously.

In this project, calibration of two mobile camera is firstly carried out based on Zhang's method. With the fixed position and attitude of cameras, the falling video of a basketball is obtained in slow motion mode. With the video processing method, high accuracy frames are derived. The position and attitude of the basketball are then achieved by 3D

reconstruction. We finally finished two different experiments, due to the place difference, we overcame the geographic restriction, successfully got two completed results.

II. METHODS

A. The theory of camera imaging

The camera imaging model is determined by the relationship between each point in the image and its corresponding point in the real space, so understanding how the camera images play an important role in algorithm building [5]. We generally refer to the camera imaging model as the pinhole model, as shown in Fig. 2

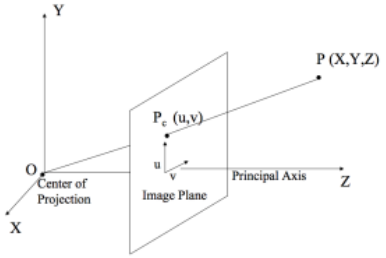

Fig. 2. Pinhole model for camera imaging

The pinhole model is mainly composed of an optical center O, principal axis XYZ, and image plane. In the pinhole model, it is assumed that the reflected beam from the surface of the object is projected onto the image plane through a pinhole according to the straight-line propagation of light. The point on the image plane is a point formed by the intersection of the line of the optical center and the point on the surface of the object and the image plane.

In the process of camera imaging, multiple coordinate systems are often involved, which are shown in Fig. 3 as follows:

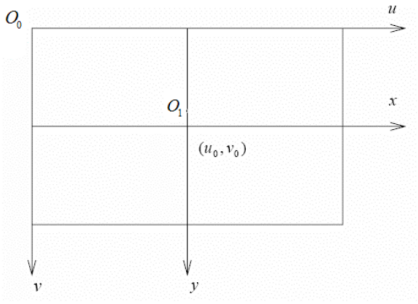

Fig. 3. The relastionship between pixel coordinate and image coordinate

The two-dimensional image data obtained by camera sampling is stored in the computer as an array. The pixel values of each image point in the array are uniformly quantized and become a finite number of discrete values. These discrete values are called the gray values of image pixels. As shown in Fig. 3, the rectangular coordinate system u-v is first defined in the imaging plane. In this coordinate system, the abscissa of an image pixel represents the number of columns of the pixel in the array, and the ordinate represents the row number of the pixel in the array, that is, (u, v) is the coordinate of the pixel in the image pixel coordinate system.

In the pixel coordinates of an image, (u, v) only represents the number of columns and rows of a pixel in the array, but does not represent the real position of the pixel in the imaging plane. So we can establish a coordinate system to represent the real coordinates of pixels on the image plane. This coordinate system is called the image physical coordinate system, as shown in Fig. 3. In the rectangular coordinate system U-V, the main point of the image, namely origin O_1, is the intersection of the optical axis of the camera and the imaging plane. Suppose that the coordinates of O_1 are (U_0, V_0), and each pixel has a certain physical size dx and dy in X and Y directions respectively, which indicates how many millimeters a pixel is. These two parameters are approximately equal, but there will be some differences due to manufacturing accuracy. Then we can get the relationship between the two coordinate systems as follows:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{dx} & 0 & u_0 \\ 0 & \frac{1}{dy} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (1)$$

where (u,v) is the coordinates under the image pixel coordinate system, (x,y) is the image physical coordinate system coordinates.

The coordinate system of an object is measured by a camera standing at its angle. The origin of the camera coordinate system is on the optical center of the camera, and the Z-axis is parallel to the optical axis of the camera. It is the bridgehead of the contact with the object to be photographed. The object in the world coordinate system needs to go through the rigid body change to the camera coordinate system, and then it has a relationship with the image coordinate system. It is the link between image coordinates and world coordinates and communicates the furthest distance in the world.

World coordinate system, as the name implies, refers to the coordinate system that provides location reference for the target object and camera in real space. The coordinate system is objective and true. We find that the homogeneous coordinates $(x_c, y_c, z_c)^T$ of the space point P in the camera coordinate system and the homogeneous coordinates $(x_w, y_w, z_w)^T$ of the point in the world system are as follows:

$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \quad (2)$$

R is the rotation matrix and T denotes the translation vector.

So the camera imaging process is to use the equation of these different systems to finally figure out the equation from the image pixel coordinate system to the world coordinate system. The camera imaging process can be divided into three steps. In the first step, the homogeneous coordinates of the space point P in the world coordinate system are transformed into the homogeneous coordinates of the point in the camera coordinate system through a rotation matrix and a translation matrix, as shown in equation 2. Where $(x_c, y_c, z_c)^T$ is the homogeneous coordinate of the camera coordinate system of space point P, and $(x_w, y_w, z_w)^T$ is the homogeneous coordinate of the point in the world coordinate system. In the second step, the homogeneous coordinates of the point in the camera coordinate system are projected onto the imaging plane through the pinhole camera model and transformed into the homogeneous coordinates of the image points in the image physical coordinate system:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \frac{1}{z_c} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} \quad (3)$$

(x, y, 1) is the homogeneous coordinate of the image point in the image physical coordinate system

Finally, the homogeneous coordinates of the image point in the image physical coordinate system and the camera parameters are transformed into the homogeneous coordinates of the image point in the image pixel coordinate system:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{dx} & 0 & u_0 \\ 0 & \frac{1}{dy} & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (4)$$

Through the above three steps, we establish the relationship between pixels and spatial points based on camera parameters. Therefore, we can get the focal length f, physical dimensions dx and dy, and the position of the main point (u_0, v_0) according to the detected point coordinates.

Generally, under the ideal premise, we can divide the camera parameters into internal and external ones: the rotation matrix and translation matrix used for the transformation from world coordinates to camera coordinates are the external parameters of the camera; the focal length f, physical dimensions dx and dy, and the main point position (u0, v0) are the camera internal parameters.

Let

$$\alpha = \frac{f}{dx} \quad (5)$$

$$\beta = \frac{f}{dy} \quad (6)$$

Combining them with the previous three stages, we can get the relationship from the world coordinate system coordinates to the image pixel coordinate system coordinates:

$$\frac{1}{z_c} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & 0 & u_0 & 0 \\ 0 & \beta & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \quad (7)$$

$$\frac{1}{z_c} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K[R][T] \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = P \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \quad (8)$$

where k is the camera internal parameter matrix and P is the projection matrix.

B. Circle detection

In order to capture the 3D coordinate of the falling ball, the 2D coordinate of ball in each camera is necessary to be obtained. In general, the centroid of the ball represents the position of the ball. Therefore, detecting the centroid of the falling ball in images in each camera is obligatory.

Hough Transformation is an extensively used feature detection method which is applied in field of image analysis, computer vision and digital image processing. The basic idea of Hough Transformation is to transform a given curve in the original image space into a point in the parameter space through the curve representation. To be specific, in circle detection, the boundary of curves in image is detected and acquired. For the points (x, y) on these curves, the circle that could pass them are represented as:

$$(x-a)^2 + (y-b)^2 = r^2 \quad (9)$$

Where a and b denote the centroid set of the circle and r is the radius of it. Therefore, for each boundary point in the image, there are a group of (a, b, r) represents the circle go through it. This group of (a, b, r) forms a cone in the parameter circle as shown in Fig.4.

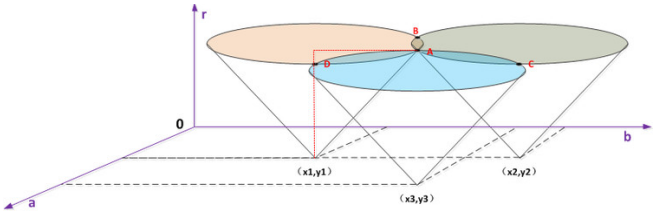

Fig. 4. The scheme of Hough Transformation circle detection

For all boundary points in the image, multiple cones are established in the parameter space. If there are circles in image, the parameter coordinate in parameter space of these points in the same circle will converge to a point in the parameter space. With more convergence of a point, the more likely there is a circle of boundary points. Hence, the circle in image is obtained by the intersection in parameter space. The centroid and radius are then acquired.

C. 3D reconstruction

With the acquired centroid coordinate, the intrinsic matrix and external matrix, the preparation of 3D reconstruction is ready. The position and attitude of the basketball could be generated afterwards. As it is depicted in equation (8), with the obtained P, u and v, the world coordinate is obtained by solving the equation:

$$\begin{cases} (u_1P_{31}^1 - P_{11}^1)x_w + (u_1P_{32}^1 - P_{12}^1)y_w + (u_1P_{33}^1 - P_{13}^1)z_w = P_{14}^1 - u_1P_{34}^1 \\ (v_1P_{31}^1 - P_{21}^1)x_w + (v_1P_{32}^1 - P_{22}^1)y_w + (v_1P_{33}^1 - P_{23}^1)z_w = P_{24}^1 - v_1P_{34}^1 \\ (u_2P_{31}^2 - P_{11}^2)x_w + (u_2P_{32}^2 - P_{12}^2)y_w + (u_2P_{33}^2 - P_{13}^2)z_w = P_{14}^2 - u_2P_{34}^2 \\ (v_2P_{31}^2 - P_{21}^2)x_w + (v_2P_{32}^2 - P_{22}^2)y_w + (v_2P_{33}^2 - P_{23}^2)z_w = P_{24}^2 - v_2P_{34}^2 \end{cases} \quad (10)$$

Since there are tiny error occurred in each step above, the result of the equation could be empty that the four lines of equation do not converge to a point, Least Square Method is applied. The result that can best fit the equation is selected as the world coordinate of the point.

III. EXPERIMENT AND RESULT

With the designed method, two experiments are conducted in this project. As it is designed above, the experiment contains calibration, video recording, video processing, circle detection, 3D reconstruction and speed and acceleration calculation.

A. Calibration

In calibration, a checkerboard is shown in the screen of MacBook Pro (13 inch). We select the screen of computer rather than a printed paper because the brightness adjustment of screen could help software better recognize the corner of the checkerboard. The slow-motion mode with 240 fps is necessary for the capture of high-speed ball and the focal length of normal video mode and slow-motion mode is different. Therefore, in order to better calculate parameters of stereo camera, in calibration and video recording, slow motion mode is applied in the whole process. Ten videos are recorded in slow-motion mode for each camera for ten different positions and attitudes of a checkerboard with grid side length

of 29.5 millimeters. The image captured from the video is shown in Fig.5.

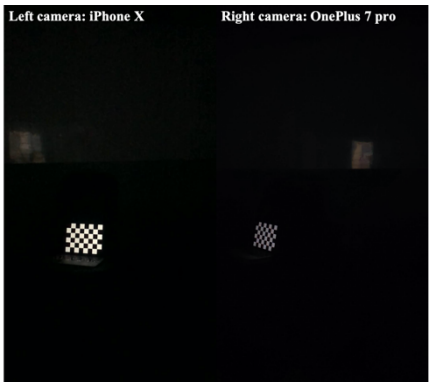

Fig. 5. The image for stereo camera calibration

As it is shown in Fig.5, the background of the video is dark and the only shining place is the screen. Therefore, the high contrast of the picture is guaranteed. The height, distance and attitude of the checkerboard are changed in each video in order to capture the distortion of cameras thoroughly in calculation.

B. *Video recording of falling ball*

With the same position and attitude of the stereo camera, the video of fall ball is recorded in slow-motion mode. The basketball is just a normal one with three markers in this surface in order to capture its attitude afterwards. The basketball is shown in Fig.6. The three markers are not parallel to the coordinate of the ball since in that way the stereo camera could not see all three markers at once.

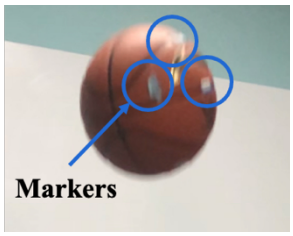

Fig. 6. Figure and markers of the basketball

The background of the video is a white-green wall with a timer in the corner as it is depicted in Fig.7. The color of the wall has a contrast with the color of the ball. Therefore, the boundary of the ball is easy to be detected. The timer in the corner which can show millisecond is utilized to facilitate the processing of video. This high accuracy timer can assist the alignment of the video in two cameras which is detailly explained in C. The basketball is throwed from a man hand with a slow speed. The ball and its markers are kept in the view of stereo camera and this needs several times try. The light in the room is also needed to be adjusted for the shadow that covers on the ball will affect the circle detection result. With all these preparation, 3 to 5 videos are recorded in slow-motion mode.

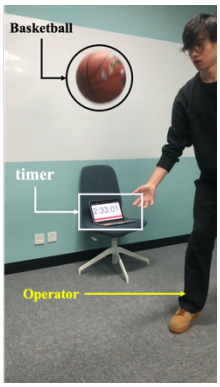

Fig. 7. The operation scene of experiments

C. *Video processing*

With the video, video processing which is consist of frame extraction and frame alignment is necessary. For calibration, since the checkerboard is static, we only randomly select a frame in the middle of the video. For tracking of a falling ball, we firstly derive all frames from the video. Secondly, with the utilization of timer, frames are selected and aligned. For example, in the Fig.8, four frames of video from left and right camera is shown. Fig.8 (a) and Fig.8 (c) is the first time in 2:33:01. Fig.8 (b) and Fig.8 (d) is the last frame in 2:33:02. Hence, with a relatively high accuracy, Fig.8 (a) and Fig.8 (c) is seemed as the same moment for the ball. Frames are selected from the time 2:33:01. 40 frames are selected for further process

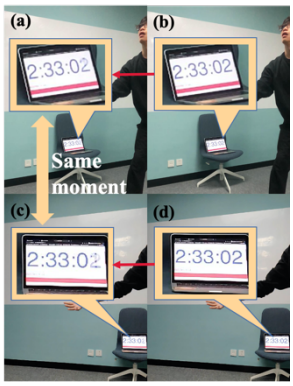

Fig. 8. The alignment process of video frames

D. *Circle detection*

Positions of the basketball in the image are detected by Hough Transformation. By inputting images into MATLAB, the circle detection is shown in Fig 9, the boundary of the basketball is recognized with a high accuracy. Result of 2D coordinate from beginning to end is generated and shown in Fig.10 (a) and (b). Fig.10 (a) and (b) are the result of left and right camera. The trajectory obtained in two figures are straight line which is consistent with our observation.

Fig. 9. The detection of ball and its centroid

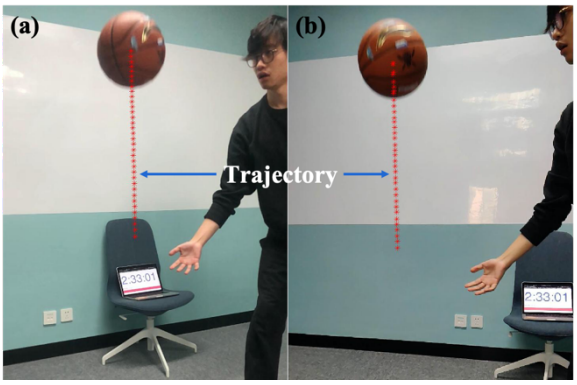

Fig. 10. The trajectory of falling ball in stereo camera

E. 3D recognition

With the 2D coordinate, intrinsic and external matrix obtained above, the 3D position and attitude of the basketball is calculated based on equation (10) and Least Square Method. The world coordinate of the falling trajectory is built in MATLAB figure Fig.11. Fig.11 (a) and (b) are the 3D reconstruction result of 2 experiments. In Fig.11 (a), the red circle is the position of basketball centroid. The green, yellow and blue arrows are the direction of three markers. In Fig.11 (b), the red circle is also the position of basketball centroid while in that experiment only 1 marker is attached to the basketball. Therefore, the yellow arrow denotes the direction of the marker. As it is depicted in Fig.11, a little rotation is observed which is coincident with the observation by our eyes. Also, the distribution of trajectory points of two figures shows the same pattern, that the distribution is compact at the beginning and sparse at the end. This phenomenon is coincident with the law of free fall that the speed of the ball becomes higher in the duration.

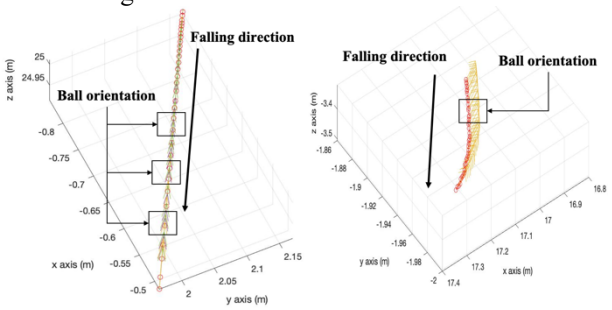

Fig. 11. Result of 3D reconstruction of two experiments

F. Velocity and acceleration

The velocity of the ball is calculated based on the 3D position distance of the ball centroid. For the time between each frame is 1/240 second, the average speed of each period is calculated as:

$$v = s/t \tag{11}$$

where v is the average velocity, s is the distance between each point and t is 1/240 second. The result of two experiments are shown in Fig.12. The beginning and end of velocity are abandoned for its inaccuracy and distortion of camera. The middle part of two curve are fitted by straight line. For the Fig.12 (a), the slope of the line is 0.0344 m/s^2, so the acceleration of this period is calculated by:

$$a = slope \times framerate \tag{12}$$

the framerate of two cameras is 240 fps. Therefore, the acceleration of experiment 1 is 8.256 m/s^2. Similarly, the acceleration of experiment 2 is 10.44 m/s^2. The average acceleration of these two experiments is 9.348 m/s^2. Since the acceleration gravity is 9.8 m/s^2, the acquired three results are very similar.

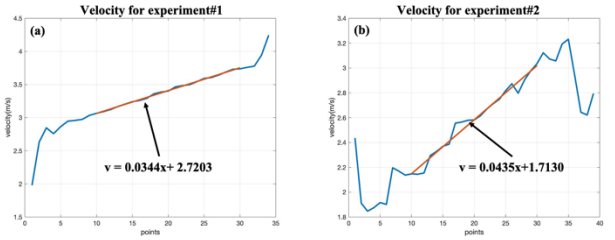

Fig. 12. The velocity and acceleration obtained in two experiments

IV. CONCLUSION AND DISCUSSION

After the text edit has been completed, the paper is ready for the template. Duplicate the template file by using the Save As command, and use the naming convention prescribed by your conference for the name of your paper. In this newly created file, highlight all of the contents and import your prepared text file. You are now ready to style your paper; use the scroll down window on the left of the MS Word Formatting toolbar.

REFERENCES

The template will number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in [3]—do not use "Ref. [3]" or "reference [3]" except at the beginning of a sentence: "Reference [3] was the first ..."

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the abstract or reference list. Use letters for table footnotes.

Unless there are six authors or more give all authors' names; do not use "et al.". Papers that have not been published, even if they have been submitted for publication, should be cited as "unpublished" [4]. Papers that have been accepted for publication should be cited as "in press" [5]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [6].

[1] G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955. *(references)*

[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[3] I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[4] K. Elissa, “Title of paper if known,” unpublished.

[5] R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.

[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[7] M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.

APPENDIX A

Code:

```
v = VideoReader('one.mp4');
numFrames = v.NumFrames;
for k = 654 : 957
        frame = read(v,k);
        a = sprintf('%03d',k);
        imwrite(frame,strcat('/Users/carl/Ottawa/3Dvision/projecton/temp/',a,'.jpg'),'jpg');
        end

imageFileNames1 = {'/Users/carl/Ottawa/3Dvision/project/ip/pic/cal001-ip.jpg',
      '/Users/carl/Ottawa/3Dvision/project/ip/pic/cal002.jpg',
      '/Users/carl/Ottawa/3Dvision/project/ip/pic/cal003.jpg',
      '/Users/carl/Ottawa/3Dvision/project/ip/pic/cal004.jpg',
      '/Users/carl/Ottawa/3Dvision/project/ip/pic/cal005.jpg',
      '/Users/carl/Ottawa/3Dvision/project/ip/pic/cal006.jpg',
      '/Users/carl/Ottawa/3Dvision/project/ip/pic/cal008.jpg',
      '/Users/carl/Ottawa/3Dvision/project/ip/pic/cal009.jpg',
      '/Users/carl/Ottawa/3Dvision/project/ip/pic/cal010.jpg',
      '/Users/carl/Ottawa/3Dvision/project/ip/pic/cal011.jpg',
      };
imageFileNames2 = {'/Users/carl/Ottawa/3Dvision/project/on/pic/cal001-on.jpg',
      '/Users/carl/Ottawa/3Dvision/project/on/pic/cal002.jpg',
      '/Users/carl/Ottawa/3Dvision/project/on/pic/cal003.jpg',
      '/Users/carl/Ottawa/3Dvision/project/on/pic/cal004.jpg',
      '/Users/carl/Ottawa/3Dvision/project/on/pic/cal005.jpg',
      '/Users/carl/Ottawa/3Dvision/project/on/pic/cal006.jpg',
      '/Users/carl/Ottawa/3Dvision/project/on/pic/cal008.jpg',
      '/Users/carl/Ottawa/3Dvision/project/on/pic/cal009.jpg',
      '/Users/carl/Ottawa/3Dvision/project/on/pic/cal010.jpg',
      '/Users/carl/Ottawa/3Dvision/project/on/pic/cal011.jpg',
      };
[imagePoints, boardSize, imagesUsed] = detectCheckerboardPoints(imageFileNames1, imageFileNames2);
squareSize = 29;
worldPoints = generateCheckerboardPoints(boardSize, squareSize);
I1 = imread(imageFileNames1{1});
[mrows, ncols, ~] = size(I1);
[stereo, pairsUsed, estimationErrors] = estimateCameraParameters(imagePoints, worldPoints, 'EstimateSkew', true,
'EstimateTangentialDistortion', true, 'NumRadialDistortionCoefficients', 2, 'WorldUnits', 'millimeters', 'InitialIntrinsicMatrix',
[], 'InitialRadialDistortion', [], 'ImageSize', [mrows, ncols]);
h1=figure; showReprojectionErrors(stereo);
h2=figure; showExtrinsics(stereo, 'CameraCentric');
displayErrors(estimationErrors, stereo);
I2 = imread(imageFileNames2{1});
[J1, J2] = rectifyStereoImages(I1, I2, stereo);

path ='/Users/carl /Ottawa/3Dvision/project /on/frames/';
Files= dir(strcat(path,'*.jpg'));
for i=1:length(Files)
      x1=Files(i).name;
      a = sprintf('%03d',i);
      x2=num2str(a);
      x3=strcat(x2,'-','on','.jpg');
      copyfile([path x1],['/Users/carl/ /Ottawa/3Dvision/project /on/ren/' x3]);
end

function [hough_space,hough_circle,para] = hough_circle(BW,step_r,step_angle,r_min,r_max,p)
circleParaXYR=[];
para=[];
[m,n] = size(BW);
size_r = round((r_max-r_min)/step_r)+1;
```

```
size_angle = round(2*pi/step_angle);
hough_space = zeros(m,n,size_r);
[rows,cols] = find(BW);
ecount = size(rows);

for i=1:ecount
    for r=1:size_r
        for k=1:size_angle
            a = round(rows(i)-(r_min+(r-1)*step_r)*cos(k*step_angle));
            b = round(cols(i)-(r_min+(r-1)*step_r)*sin(k*step_angle));
            if(a>0.35*m&a<=0.7*m&b>0.35*n&b<=0.7*n)
                hough_space(a,b,r) = hough_space(a,b,r)+1;
            end
        end
    end
end
max_para = max(max(max(hough_space)));
index = find(hough_space>=max_para*p);
length = size(index);
hough_circle = false(m,n);
for i=1:ecount
    for k=1:length
        par3 = floor(index(k)/(m*n))+1;
        par2 = floor((index(k)-(par3-1)*(m*n))/m)+1;
        par1 = index(k)-(par3-1)*(m*n)-(par2-1)*m;
        if((rows(i)-par1)^2+(cols(i)-par2)^2<(r_min+(par3-1)*step_r)^2+5&...
                (rows(i)-par1)^2+(cols(i)-par2)^2>(r_min+(par3-1)*step_r)^2-5)
            hough_circle(rows(i),cols(i)) = true;
        end
    end
end
for k=1:length
    par3 = floor(index(k)/(m*n))+1;
    par2 = floor((index(k)-(par3-1)*(m*n))/m)+1;
    par1 = index(k)-(par3-1)*(m*n)-(par2-1)*m;
    circleParaXYR = [circleParaXYR;par1,par2,par3];
    hough_circle(par1,par2)= true;
    %fprintf(1,'test1:Center %d %d \n',par1,par2);
end
while size(circleParaXYR,1) >= 1
    num=1;
    XYR=[];
    temp1=circleParaXYR(1,1);
    temp2=circleParaXYR(1,2);
    temp3=circleParaXYR(1,3);
    c1=temp1;
    c2=temp2;
    c3=temp3;
    temp3= r_min+(temp3-1)*step_r;
    if size(circleParaXYR,1)>1
        for k=2:size(circleParaXYR,1)
          if (circleParaXYR(k,1)-temp1)^2+(circleParaXYR(k,2)-temp2)^2 > temp3^2
              XYR=[XYR;circleParaXYR(k,1),circleParaXYR(k,2),circleParaXYR(k,3)];
          else
            c1=c1+circleParaXYR(k,1);
            c2=c2+circleParaXYR(k,2);
            c3=c3+circleParaXYR(k,3);
            num=num+1;
          end
        end
    end
    %fprintf(1,'sum %d %d radius %d\n',c1,c2,r_min+(c3-1)*step_r);
```

```
        c1=round(c1/num);
        c2=round(c2/num);
        c3=round(c3/num);
        c3=r_min+(c3-1)*step_r;
        %fprintf(1,'num=%d\n',num)
        %fprintf(1,'Center %d %d radius %d\n',c1,c2,c3);
        para=[para;c1,c2,c3];
        circleParaXYR=XYR;
end

function [position_d] = big_find_circle_d()
path ='/Users/carl/ Ottawa/3Dvision/project/on/ren/';
Files= dir(strcat(path,'*.jpg'));
cen_x = zeros(length(Files),1);
cen_y = zeros(length(Files),1);
xx_ = 400;
yy_ = 226;
for i=1:44
        [xx,yy] = testcircle_d(Files(i).name,0.99,xx_,yy_);
        cen_x(i) = xx;
        cen_y(i) = yy;
        xx_ = xx;
        yy_ = yy;
end
position_d = [cen_x(1:44),cen_y(1:44)];

I = imread('151-ip.jpg');
h = rgb2gray(I);
h = imcomplement(h);
h = medfilt2(h,[4,4]);%[4,4]
bw = im2bw(h,graythresh(h));
se = strel('disk',2);
bw = imclose(bw,se);
figure;imshow(bw);
bw    = imfill(bw,'holes');
[B,L] = bwboundaries(bw,'noholes');
figure;imshow(label2rgb(L, @jet, [.5 .5 .5]))
hold on
gray_image = rgb2gray(label2rgb(L, @jet, [.5 .5 .5]));
imshow(gray_image)
[centers,radii] = imfindcircles(gray_image,[100 140],'ObjectPolarity','dark','Sensitivity',0.975);
imshow(rgb)
h = viscircles(centers,radii);

position_e = big_find_circle_e();
position_d = big_find_circle_d();
rgb1 = imread('057-ip.jpg');
figure(1)
imshow(rgb1)
hold on
x_e = position_e(1:35,1);
y_e = position_e(1:35,2);
plot(x_e,y_e,'r*');

figure(2)
rgb2 = imread('057-on.jpg');
imshow(rgb2)
hold on
x_d = position_d(1:35,1);
y_d = position_d(1:35,2);
plot(x_d,y_d,'r*');
```

```
att_on_l = zeros(50,2);
for i=1:50
	ind = i+56;
	if ind>99
		ind = int2str(i+56);
	else
		ind = strcat('0',int2str(ind));
	end
	name = strcat(ind,'-on.jpg');
	imageMatrix1 = imread(name,'jpg');
	imagesc(imageMatrix1);
	axis('equal');
	[u1,v1] = ginput(1);
	att_on_l(i,:) = [u1,v1];
end

att_on_m = zeros(50,2);
for i=1:50
	ind = i+56;
	if ind>99
		ind = int2str(i+56);
	else
		ind = strcat('0',int2str(ind));
	end
	name = strcat(ind,'-on.jpg');
	imageMatrix1 = imread(name,'jpg');
	imagesc(imageMatrix1);
	axis('equal');
	[u1,v1] = ginput(1);
	att_on_m(i,:) = [u1,v1];
end

att_on_r = zeros(50,2);
for i=1:50
	ind = i+56;
	if ind>99
		ind = int2str(i+56);
	else
		ind = strcat('0',int2str(ind));
	end
	name = strcat(ind,'-on.jpg');
	imageMatrix1 = imread(name,'jpg');
	imagesc(imageMatrix1);
	axis('equal');
	[u1,v1] = ginput(1);
	att_on_r(i,:) = [u1,v1];
end

in_p = parameter.intrinsic01;
in_o = parameter.intrinsic02;
R_1_1 = parameter.rotation01(:,:,1);
R_1_2 = parameter.rotation02(:,:,1);
T_1_1 = parameter.trans01(1,:)';
T_1_2 = parameter.trans02(1,:)';
M1 = in_p*[R_1_1 T_1_1];
M2 = in_o*[R_1_2 T_1_2];
%i = 5;
%syms x y z;
%e1 = (x_e(i)*M1(3,1)-M1(1,1))*x+(x_e(i)*M1(3,2)-M1(1,2))*y+(x_e(i)*M1(3,3)-M1(1,3))*z-M1(1,4)+x_e(i)*M1(3,4);
%e2 = (y_e(i)*M1(3,1)-M1(2,1))*x+(y_e(i)*M1(3,2)-M1(2,2))*y+(y_e(i)*M1(3,3)-M1(2,3))*z-M1(2,4)+y_e(i)*M1(3,4);
%e3 = (x_d(i)*M2(3,1)-M2(1,1))*x+(x_d(i)*M2(3,2)-M2(1,2))*y+(x_d(i)*M2(3,3)-M2(1,3))*z-M2(1,4)+x_d(i)*M2(3,4);
%e4 = (y_d(i)*M2(3,1)-M2(2,1))*x+(y_d(i)*M2(3,2)-M2(2,2))*y+(y_d(i)*M2(3,3)-M2(2,3))*z-M2(2,4)+y_d(i)*M2(3,4);
```

```
%[x0 y0 z0] = solve(e3,e4,x,y,z)
x_e = position_e(9:40,1);
y_e = position_e(9:40,2);
x_d = position_d(9:40,1);
y_d = position_d(9:40,2);
x = zeros(32,3);
y1 = zeros(32,3);
y2 = zeros(32,3);
y3 = zeros(32,3);
sf = 46*30/100;
for i = 1:32
    A = [(x_e(i)*M1(3,1)-M1(1,1)) (x_e(i)*M1(3,2)-M1(1,2)) (x_e(i)*M1(3,3)-M1(1,3))
    (y_e(i)*M1(3,1)-M1(2,1)) (y_e(i)*M1(3,2)-M1(2,2)) (y_e(i)*M1(3,3)-M1(2,3))
    (x_d(i)*M2(3,1)-M2(1,1)) (x_d(i)*M2(3,2)-M2(1,2)) (x_d(i)*M2(3,3)-M2(1,3))
    (y_d(i)*M2(3,1)-M2(2,1)) (y_d(i)*M2(3,2)-M2(2,2)) (y_d(i)*M2(3,3)-M2(2,3))];
    B = [M1(1,4)-x_e(i)*M1(3,4)
    M1(2,4)-y_e(i)*M1(3,4)
    M2(1,4)-x_d(i)*M2(3,4)
    M2(2,4)-y_d(i)*M2(3,4)];
    m=A'*B;
    n=A'*A;
    x(i,:)=n\m*sf;
end
x_e = att_ip_l(9:40,1);
y_e = att_ip_l(9:40,2);
x_d = att_on_l(9:40,1);
y_d = att_on_l(9:40,2);
for i = 1:32
    A = [(x_e(i)*M1(3,1)-M1(1,1)) (x_e(i)*M1(3,2)-M1(1,2)) (x_e(i)*M1(3,3)-M1(1,3))
    (y_e(i)*M1(3,1)-M1(2,1)) (y_e(i)*M1(3,2)-M1(2,2)) (y_e(i)*M1(3,3)-M1(2,3))
    (x_d(i)*M2(3,1)-M2(1,1)) (x_d(i)*M2(3,2)-M2(1,2)) (x_d(i)*M2(3,3)-M2(1,3))
    (y_d(i)*M2(3,1)-M2(2,1)) (y_d(i)*M2(3,2)-M2(2,2)) (y_d(i)*M2(3,3)-M2(2,3))];
    B = [M1(1,4)-x_e(i)*M1(3,4)
    M1(2,4)-y_e(i)*M1(3,4)
    M2(1,4)-x_d(i)*M2(3,4)
    M2(2,4)-y_d(i)*M2(3,4)];
    m=A'*B;
    n=A'*A;
    y1(i,:)=n\m*sf;
end
x_e = att_ip_m(9:40,1);
y_e = att_ip_m(9:40,2);
x_d = att_on_m(9:40,1);
y_d = att_on_m(9:40,2);
for i = 1:32
    A = [(x_e(i)*M1(3,1)-M1(1,1)) (x_e(i)*M1(3,2)-M1(1,2)) (x_e(i)*M1(3,3)-M1(1,3))
    (y_e(i)*M1(3,1)-M1(2,1)) (y_e(i)*M1(3,2)-M1(2,2)) (y_e(i)*M1(3,3)-M1(2,3))
    (x_d(i)*M2(3,1)-M2(1,1)) (x_d(i)*M2(3,2)-M2(1,2)) (x_d(i)*M2(3,3)-M2(1,3))
    (y_d(i)*M2(3,1)-M2(2,1)) (y_d(i)*M2(3,2)-M2(2,2)) (y_d(i)*M2(3,3)-M2(2,3))];
    B = [M1(1,4)-x_e(i)*M1(3,4)
    M1(2,4)-y_e(i)*M1(3,4)
    M2(1,4)-x_d(i)*M2(3,4)
    M2(2,4)-y_d(i)*M2(3,4)];
    m=A'*B;
    n=A'*A;
    y2(i,:)=n\m*sf;
end
x_e = att_ip_r(9:40,1);
y_e = att_ip_r(9:40,2);
x_d = att_on_r(9:40,1);
y_d = att_on_r(9:40,2);
for i = 1:32
```

```
        A = [(x_e(i)*M1(3,1)-M1(1,1)) (x_e(i)*M1(3,2)-M1(1,2)) (x_e(i)*M1(3,3)-M1(1,3))
        (y_e(i)*M1(3,1)-M1(2,1)) (y_e(i)*M1(3,2)-M1(2,2)) (y_e(i)*M1(3,3)-M1(2,3))
        (x_d(i)*M2(3,1)-M2(1,1)) (x_d(i)*M2(3,2)-M2(1,2)) (x_d(i)*M2(3,3)-M2(1,3))
        (y_d(i)*M2(3,1)-M2(2,1)) (y_d(i)*M2(3,2)-M2(2,2)) (y_d(i)*M2(3,3)-M2(2,3))];
        B = [M1(1,4)-x_e(i)*M1(3,4)
        M1(2,4)-y_e(i)*M1(3,4)
        M2(1,4)-x_d(i)*M2(3,4)
        M2(2,4)-y_d(i)*M2(3,4)];
        m=A'*B;
        n=A'*A;
        y3(i,:)=n\m*sf;
end
figure(1)

scatter3(x(:,1),x(:,2),x(:,3),'ro');
hold on
plot3(x(:,1),x(:,2),x(:,3),'r-');
axis('equal')
hold on
xlabel('x axis (m)');
ylabel('y axis (m)');
zlabel('z axis (m)');
quiver3(x(:,1),x(:,2),x(:,3),-y1(:,1)+x(:,1),-y1(:,2)+x(:,2),-y1(:,3)+x(:,3));
hold on
quiver3(x(:,1),x(:,2),x(:,3),-y2(:,1)+x(:,1),-y2(:,2)+x(:,2),-y2(:,3)+x(:,3));
hold on
quiver3(x(:,1),x(:,2),x(:,3),-y3(:,1)+x(:,1),-y3(:,2)+x(:,2),-y3(:,3)+x(:,3));

v_a_y = zeros(34,2);
dista_2_y = zeros(34,1);
for i = 2:35
        dista = sqrt((point3d(i,1)-point3d(i-1,1))^2+(point3d(i,2)-point3d(i-1,2))^2+(point3d(i,3)-point3d(i-
1,3))^2);%point3d(i,3)-point3d(i-1,3);
        dista_2_y(i-1) = dista;
        %t_ = dista/(1/240*9.8)+1/240;
        %t_2(i-1) = t_;
        vv = dista*120;
        v_a_y(i-1,1) = vv;
end
for i = 4:24
        acc = (va_y(i)-va_y(i-1))*240;
        v_a_y(i,2) = acc;
end
figure(1)
plot(v_a_y(1:25,1)/1000);
xlabel('points');
ylabel('velocity(m/s)');
grid on;
figure(2)
plot(v_a_y(1:24,2)/1000);
xlabel('points');
ylabel('acceleration(m/s2)');
grid on

imageMatrix1 = imread('cal001-ip.jpg','jpg');
imageMatrix2 = imread('cal001-on.jpg','jpg');

imagesc(imageMatrix1);
axis('equal');
[u1,v1] = ginput(1);
imagesc(imageMatrix2);
axis('equal');
```

```
[u2,v2] = ginput(1);
x = zeros(1,3);
A = [(u1*M1(3,1)-M1(1,1)) (u1*M1(3,2)-M1(1,2)) (u1*M1(3,3)-M1(1,3))
      (v1*M1(3,1)-M1(2,1)) (v1*M1(3,2)-M1(2,2)) (v1*M1(3,3)-M1(2,3))
      (u2*M2(3,1)-M2(1,1)) (u2*M2(3,2)-M2(1,2)) (u2*M2(3,3)-M2(1,3))
      (v2*M2(3,1)-M2(2,1)) (v2*M2(3,2)-M2(2,2)) (v2*M2(3,3)-M2(2,3))];
B = [M1(1,4)-u1*M1(3,4)
      M1(2,4)-v1*M1(3,4)
      M2(1,4)-u2*M2(3,4)
      M2(2,4)-v2*M2(3,4)];
m=A'*B;
n=A'*A;
x=n\m;
sf = 29/sqrt(x(1)^2+x(2)^2+x(3)^2)*3
```