Multi-sensor 3D Geometry

By Celia Lewis, ECE '18

Abstract

This article will discuss the purpose of 3D geometry used in computer vision (CV) systems along with different issues that arise in CV systems and their 3-D geometry solutions counterparts. This article will also touch upon how 3-D geometry and coordinate transforms are applied to real examples along with how they are used in my senior design project.

Introduction

The way humans see and perceive three-dimensional objects is with such ease that it does not even register in a human's mind that anything out of the ordinary is happening. Computer vision's main purpose is replicate how humans detect these objects. "Researchers in computer vision have been developing [ ] mathematical techniques for recovering the three-dimensional shape and appearance of objects in imagery." (Szeliski, 3) Yet, computer vision hardly compares to human sight, which begs the question of why is vision so hard for a computer to replicate? This answer lies in the fact that "we seek to recover some unknowns given insufficient information to fully specify the solution." (Szeliski, 3) Using the computer, we are attempting to rebuild 3-D reality with 2-D imaging. The best solutions for computer vision therefore start with understanding three-dimensional geometry and how this can map into two-dimensional geometry and vice versa. With this lies the basis of all computer vision: multi-sensor three-dimensional geometry.

The basis of 3-D geometry starts with the simple problem of correspondence. That is for two images taken from two cameras, which pixel in one image lines up with the pixel in the other? In the simplest

of pixels to the right set. The solution to finding which pixel in the right image matches a pixel in the left image starts with finding the distance between a point in the left image minus a point in the right image, called the disparity (X_L-X_R). The corresponding point in a right image given a point in the left image must lie on the same scanline, meaning the vertical distance y for the point in the left image is the same as the vertical distance y for the point in the right image. This is called the epipolar constraint.

To determine the true correspondence point in the right image for the point in the left image; constraints are placed on the problem to help narrow down the exact pixel. First is the Cheirality Constraint: $X_L \geq X_R$. If $X_L < X_R$, this would make a negative disparity, meaning the object we are looking at would be behind the camera, which is clearly not possible. Next is the Maximum Disparity Constraint; this enforces a minimum distance from the camera to the surface being viewed. Following is the Uniqueness Constraint; if X_L and X_R are a match, there are no other points in the right image that match with that specific point in the left image and vice versa. Finally, the Ordering Constraint; if X_{L1} and X_{R1} are a match and X_{L2} is less than X_{L1}, the corresponding matching point of $X_{L2} = X_{R2}$, must be less than X_{R1}. (Birchfield, 626)

Once the constraints are laid out, the actual algorithm to determining which pixel from one image matches the other can be written. The most basic algorithm is known as the Block Matching Algorithm. "Block matching is an area-based [solution] that relies upon statistical correlation

in the left image, the right image is searched for the best match among all possible disparities." (Birchfield, 629) The best disparity is the one that gives the lowest sum of dissimilarities over a disparity. This is the most basic algorithm for stereo matching; however, other solutions improve running time and use Block Matching as the basis for a faster and more efficient solution.

However, the two cameras are not rectified. The algorithm for stereo matching will become more or less unusable and a new algorithm must be implemented that takes into account rotations to solve 3-D geometry and coordinate transformation problems that arise.

Rotations in 3-D Space:

Understanding any rotation is beginning with knowing the rotation matrix R. A rotation matrix takes a set of defined axis, and rotates them a certain amount to define a new set of axis pointing in a new direction. Imagine a typical x-axis and y-axis defined on a sheet of paper where the origin is bottom left corner, the y-axis is the long side of the paper straight up and the x-axis lies on the short side of the paper to the right. Now rotate the entire sheet of paper 45 degrees clockwise. How does one define where the axis lies with respect to the old axis? This is what rotation matrices do. "A rotation is a linear transformation R, that fixes the origin, preserves the length of vectors, and preserves the orientation of bases." (Heard, 7) Diving deeper into the math of matrices, it is proven that the set of all eigenvectors corresponding to the eigenvalue of R, which is equal to one, forms the axis of rotation. As a result, the simplest way to apply R is using Plane rotations.

In Plane rotations, let the vector represented as coordinates (x,y) be a complex number $x + iy = |x + y| * e^{i\theta}$; then, a counterclockwise rotation would shift the phase by some angle ϕ, making the rotated vector be $|x + y| * e^{i(\theta+\phi)}$. This vector in rectangular coordinates is $x * cos\theta - y * sin\theta + i(x * sin\theta + y * cos\theta)$ (Heard, 8). This gives rise to the three basic rotation matrices in x,y, and z space.

These matrices represent the rotation by angle θ about the x,y,z axes respectively.

$$R_x(\theta)= \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

$$R_y(\theta)= \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

$$R_z(\theta)= \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(Heard, 8)

The general rotation operator is thus defined by N, a matrix given, and an angle θ to be $R = I + sin\theta N + (1 - cos\theta)N^2$ (Heard, 10).

Rotations and the rotation matrix can be parameterized in many different ways, each for unique purposes. For example, the Euler angle parameterization, made up of three intermediate rotations, is used in aeronautics and astronautics where the three intermediate rotations are called yaw, pitch, and roll. These define which way an air vehicle is rotating. The list of parameterization of 3-D rotations for specific applications also includes quaternions, which is the best parameterization for computer vision purposes.

Example

Determining Human Height:

While many methods exist to determine human height in images, the solution below uses coordinate transformations and 3-D geometry, which does not need a real reference plane, and allows the camera to be easily rotated or translated while maintaining important information obtained from the previous position of the camera. The base of this solution uses the intrinsic parameters of the camera to "transform the image coordinate system to the camera reference frame." (Zhou, 718) The two other big factors that go into this are the angle between the camera's optical center and ground and the height of the camera from the ground.

The first coordinates to determine before beginning a coordinate transformation are the height of the person and the distance from the camera that they stand.

When the camera's optical axis is parallel to the ground and the person is perpendicular to the ground, *TB* in the figure below represents the person in the view of the camera, *H* is the height, *D* is the distance between the camera and the person, and *tb* is the image of the person *TB* on the imaging plane.

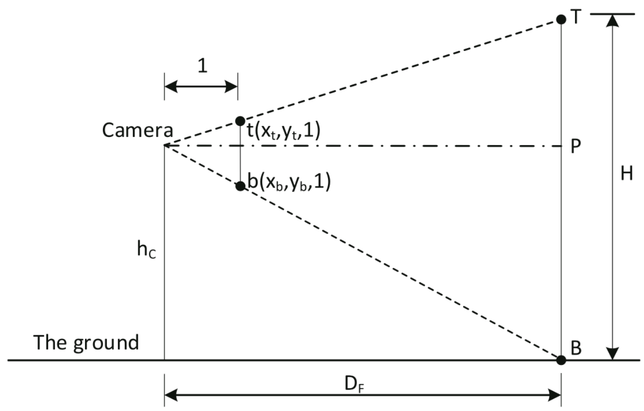

Fig. 1. The front view of the geometric model of the real scene

(Zhou, 719).

Letting the optical axis of the camera be the z-axis, the points of t and b are shown above. Then by using similar triangles theorem, the height of the person is

$\mathrm{H} = \mathrm{h}_c(1 - \frac{y_t}{y_b})$, with h_c as the height of the camera. To find the distance between the camera and the person (D_f), the computation is as follows: $D_f = \frac{h_c}{y_b}$ (Zhou, 719).

Once the height and distance are found, the next step is to determine the three-dimensional coordinates of *t* and *b* in the camera reference. This is not possible directly from the image; thus, where coordinate transformations come into the equation. The intrinsic parameters of the camera are necessary for coordinate transforms because they "establish the relationship between the points in the camera reference frame and the pixel coordinates of the points on the images captured from the camera." (Zhou, 720)

In the image frame of reference, *t* coordinates are denoted as (u_t,v_t); so the coordinates in the camera reference frame for *t*, $(x_1,y_1,1)$, as

estimated by:

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = A^{-1} \begin{bmatrix} u_t \\ v_t \\ 1 \end{bmatrix}$$

(Zhou, 720)

Where A is the intrinsic parameter (intrinsic parameters are characteristics of the specific camera used) matrix given by:

$$A = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

(Zhou, 720)

where f_x and f_y are the focal lengths in the x and y direction and u_0 and v_0 are the principal points in the x and y direction. A^{-1} can also be used to compute the coordinates of *b* in exactly the same manner it was used to find the coordinates of *t*.

This solution is given based on the idea that there is no rotation on the camera. Given that the camera is not parallel with the ground, there exists an angle β between the optical axis and ground. In order to use the coordinate transformations above, β must be taken into account in order to rotate the current reference frame to the standard camera coordinate system. The rotation matrix *R* is used in the below transformation in order to adjust the rotated camera to lessen the angle β. Refer to the $R_x(\theta)$ as *R*.

to adjust the coordinates.

$$\lambda_1 \begin{bmatrix} x_t \\ y_t \\ 1 \end{bmatrix} = R \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

(Zhou, 720)

The coordinates achieved from the above equation are normalized in the z-axis. After these two transformations, the normalized three-dimensional coordinates exist for *t* and *b*. As long as the camera's height, h_c, is known, then these three pieces of information is all that is needed to determine a person's height. The height of the camera can then be estimated by putting a reference object with a known height in the front of the camera, h_c. The equation then to estimate h_c is as follows:

$$h_c = \frac{y_b}{y_b - y_t} h_R$$

(Zhou, 721).

Application

RF Tracking UAV for Sports Video Capturing:

The overarching reason for understanding 3-D geometry at its core along with different applications is to be able to apply it to my senior design project. My senior design project is to track an athlete and their movements during a sports game using an autonomous drone. While there are several different subsystems that are required to work to achieve this goal, coordinate transformations and rotational matrices are essential when finding the object of interest in the video cameras' frame. The first step in identifying that a certain set of pixels in our image maps to the person we are looking for on the field below was to understand and map where the camera was in terms of the fields point of view. Imagine you are a bug on the middle of the Tufts lacrosse field, Bello Field. You look up and notice a camera above you, the camera has a GPS tracking system embedded in it so the camera knows where itself is in the world and thus the camera knows where is it relative to the field, but you as a bug have no GPS. How do you figure out where the camera is in terms of where you are?

To do this, you must find a rotation matrix that takes the camera's coordinates and rotates

them to your coordinates. This was exactly the first step my group took when tackling this problem.

Once we could map the camera on the field's plane and the field on the camera's plane via rotation matrices, the next step was to determine where a specific person was in the frame of the camera. In order to do this, we imagined the camera having a laser pointing directly out of its lens to the field. This laser beam is exactly over the person we are searching for, goes straight through that person and hits the field at a certain point. This point, the intersection between the field's plane and the ray of the camera, has to be defined in order to map a person to pixels in the camera. In order to do this, we had to parameterize a ray and find its intersection with a plane. Since this equation is already well defined, our job was to take existing information and combine our knowledge of our specific camera and 3-D geometry, and create our specific equation to find this intersection point. Utilizing 3-D geometry and its existing applications, my senior design group was able to locate a moving person in a video camera frame successfully.

Conclusion

3-D geometry is essential to the creation and implemented solutions of computer vision in the real world. It makes computer vision more modular by allowing the camera be at any angle is wants and providing the necessary coordinate transformations to map a 3-D world into a 2-D image or video. Computer vision is being used today in a variety of different applications such as medical imaging, surveillance, and fingerprint recognition. Computer vision has vastly improved different existing technologies; and without 3-D geometry, this would not have been possible.

References

1 Birchfield, S. (2018). *Image Processing and Analysis*. Boston, MA: Cengage Learning.

2. Deng, X., Wan, X., Zhang, Z., Leng, B., Lou, N., & He, S. (2010). Multi-camera calibration based on openCV and multi-view registration. *Proceedings of SPIE - The International Society for Optical Engineering, 7656*, Proceedings of SPIE - The International Society for Optical Engineering, Vol.7656.

3. Heard, W. B. (2006). *Rigid Body Mechanics: Mathematics, Physics and Applications*. Weinheim: Wiley-VCH.

4. Szeliski, R. (2010). *Computer Vision: Algorithms and Applications*. Springer.

5. Zhou, X., Jiang, P., Zhang, X., Zhang, B., & Wang, F. (2016). The measurement of human height based on coordinate transformation. *Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in*

Bioinformatics), 9773, 717-72