Traffic Light Triangulation Problem - Iterative Non-Linear Least Squares and the Rays Approach

Catherine Glossop

August 2020

1 Introduction

The basics of this task are to take images of an object collected from the blackfly camera feed of a rosbag and convert the pixel coordinate centres of this object to a single world coordinate point that precisely and accurately identifies the ground truth location. The application of this algorithm is to locate objects, namely traffic lights, that haven't been included in the M city map based on data collected from a rosbag. This can alternatively be used to build a basic map of key objects in the car's environment.

The first approach to a triangulation algorithm for this problem, iterative non-linear least squares consists of a projection of 2D pixel coordinates to 3D world coordinates using a homogeneous camera matrix. This transformation is performed for a series of images that capture the object and iterative non-linear least squares is used to approach the 3D world coordinate solution that minimizes the residual error.

The second approach, the 3D rays approach, consists of a projection of 2D pixel coordinates to 3D world coordinates using 3D rays that connect the camera centre to the 2D pixel coordinate object centre. This is performed for each sample image and the point that has the minimal distance between all itself and it's projection into the sample images rays is determined to be the best estimate of the 3D world coordinate location of the object centre.

Each of these methods will be outlined and then a brief comparison will be given.

2 Iterative Non-linear Least Squares

To be able to get accurate results, a single calculation based on the collected information is not always enough. Rather iterating and descending to the solution that has the lowest cost is the best option, and for that iterative least squares is used. Equation 1 is minimizes to minimize the squared error between

the collected object centres and the actual location of the object in 2D space.

$$E_{NLS}(\mathbf{\Delta p}) = \sum_i \|\mathbf{f}(\mathbf{p} + \mathbf{\Delta p}; \mathbf{P}) - \mathbf{x'_i}\|^2 \tag{1}$$

For this approach, there is a set of 2D object centres [1] measured and collected from the i camera images pulled from the rosbag, denoted as $\mathbf{x'_i} = (u'_i, v'_i)$. The current approximation of the 3D world coordinate location of the object's centre is denoted as $\mathbf{p}$ = (X,Y,Z,W). This approximation is updated by Δp with each iteration. The function $\mathbf{f}$ transforms $\mathbf{p}$ to a predicted object centre position, $\mathbf{x_i} = (u_i, v_i)$ in 2D camera coordinates. The details of how this is achieved will be discussed in section 3. The goal of Equation 1 is to minimize the squared difference between the measured and predicted 2D camera coordinate object centres and find the $\mathbf{\Delta p}$ that will further minimize this error, eventually getting to the best estimate of the 3D world position given our known information. The next step is to break down Equation 1 further so we can successfully extract $\mathbf{\Delta p}$. To do this, we estimate Equation 1 as

$$E_{NLS}(\mathbf{\Delta p}) \approx \sum_i \|\mathbf{J}(\mathbf{p} + \mathbf{\Delta p}; \mathbf{P})\mathbf{\Delta p} - \mathbf{r_i}\|^2 \tag{2}$$

Where $\mathbf{J}$ is the Jacobian of $\mathbf{f}(\mathbf{x_i}; \mathbf{p})$ which can also be denoted as $\frac{d\mathbf{f}}{d\mathbf{p}}$ or $\frac{\partial(u_i, v_i)}{\partial(X,Y,Z,W)}$. The residual, $\mathbf{r_i}$ is difference between the predicted and measured camera coordinate centres, $\mathbf{r_i} = \mathbf{f}(\mathbf{x_i}; \mathbf{p}) - \mathbf{x'_i}$. Through a series of simplifications, a form of Equation 2 can be achieved that allows us solve for $\mathbf{\Delta p}$. This form is shown in Equation 3.

$$(\mathbf{A} + \lambda diag(\mathbf{A}))\mathbf{\Delta p} = \mathbf{b} \tag{3}$$

where

$$\mathbf{A} = \sum_i \mathbf{J^T J} \tag{4}$$

and

$$\mathbf{b} = \sum_i \mathbf{J^T r_i} \tag{5}$$

and λ is a positive factor which allows us to adjust the size that we are stepping with. A small value decreases step size but will increase the time for the algorithm to converge. A large value will decrease the time to converge but may cause the minimum to be missed because the steps are too large. Using these equations, the general process is as follows:

1. Give an initial guess for $\mathbf{p}$. This can be estimated or linear least squares can be used to give a guess.

[1] These centres are computed by selecting two points on the image of a rectangle bounding the object, one at the top left corner, $\mathbf{p_1} = (p_x^{(1)}, p_y^{(1)})$, and one at the bottom right corner, $\mathbf{p_2} = (p_x^{(2)}, p_y^{(2)})$. The centre is computed as $\mathbf{x'_i} = (\frac{(p_x^{(2)} - p_x^{(1)})}{2}, \frac{(p_y^{(2)} - p_y^{(1)})}{2})$

2. Each camera image from the rosbag will have a different J (explained in section 3). Using $\mathbf{p}$, calculate each term of $\mathbf{A}$ and $\mathbf{b}$ for each image.

3. Sum up these terms to get $\mathbf{A}$ and $\mathbf{b}$ for the entire problem, as shown in Equations 4 and 5.

4. Use Equation 3 to solve for $\mathbf{\Delta p}$.

5. Update $\mathbf{p} \leftarrow \mathbf{p} + \mathbf{\Delta p}$

6. Repeat steps 2 to 5 until a desired threshold is reached or a maximum number of iterations in performed. There are also cases in which the algorithm will not converge.

Now, there is a good understanding of how we will be able to get our final solution, $\mathbf{p}$. Next, $\mathbf{f}(\mathbf{x_i}; \mathbf{p})$, the function for transforming 2D camera coordinates to 3D world coordinates, or the projection function, will be discussed.

The groundwork has been laid and now we can begin get into the specifics of the triangulation problem. The first set of equations that is required are those equations that transform 3D world coordinate points to 2D camera coordinate points, denoted as $\mathbf{f}$ in Equation 1.

2.1 Triangulation Equations

This function outputs $\mathbf{x_i} = (u_i, v_i)$, the predicted camera coordinate object centres. A simple formulation to get this function is by using the $\mathbf{P_i}$, the 4 x 4 camera matrix, to project the points, shown in Equations 6 and 7. Note that $\mathbf{P_i}$ will be different for each of the i images that are being observed and therefore, although $\mathbf{P_i}$ is known, it will change for each $\mathbf{x_i}$.

$$u_i = \frac{p_{00}^{(i)} X + p_{01}^{(i)} Y + p_{02}^{(i)} Z + p_{03}^{(i)} W}{p_{20}^{(i)} X + p_{21}^{(i)} Y + p_{22}^{(i)} Z + p_{23}^{(i)} W} \tag{6}$$

$$v_i = \frac{p_{10}^{(i)} X + p_{11}^{(i)} Y + p_{12}^{(i)} Z + p_{13}^{(i)} W}{p_{20}^{(i)} X + p_{21}^{(i)} Y + p_{22}^{(i)} Z + p_{23}^{(i)} W} \tag{7}$$

Now there is the problem of determining what the matrix $\mathbf{P_i}$ is. This will be described in section 3.2.

2.2 The Camera Matrix

The general formulation for the camera matrix is the multiplication of the 4 x 4 camera intrinsic matrix, $\mathbf{K}$, and a 4 x 4 transformation matrix, $\mathbf{E}$ between the two frames of interest.

$$\mathbf{P} = \mathbf{KE} \tag{8}$$

This matrix can be used to transform from the pixel coordinates to world coordinates.

$$\mathbf{x_i'} = \mathbf{Pp} \tag{9}$$

where $\mathbf{x_i'}$ is the pixel coordinate point equivalent to $\mathbf{p}$ in disparity form.[2]. The 4 x 4 camera intrinsic matrix is obtained from calibration techniques and remains fixed for the same camera for all the images of interest. In this case, the camera intrinsic matrix is as follows:

$$\mathbf{K} = \begin{bmatrix} 1449.144043 & 0 & 1199.158041 & 0 \\ 0 & 1456.590741 & 1036.123288 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{10}$$

This is simple enough to use once we've got it. The more difficult portion of $\mathbf{P}$ to extract is the transformation matrix, $\mathbf{E}$. This transformation matrix must take the points from the odometry frame, called "odom" in the zeus repository and bring it to the camera frame, called "monolink" in the zeus repository. This transformation is broken down as follows:

$$\mathbf{E} = \mathbf{T_{co}} = \mathbf{T_{ci}}\mathbf{T_{io}} \tag{11}$$

Where the first letter in the subscript denotes the frame the point is being transformed to and the second letter denotes the frame the point is currently in. Therefore, to transform from the odometry frame to the camera frame, we deconstruct the transformation matrix into a transformation from "odom" frame to the "imulink" frame, which is more readily available, and a transformation from the "imulink" frame to the "monolink" frame.

The $\mathbf{T_{io}}$ matrix can be extracted indirectly from the "get_odom_tf" function in the "transform_utils.cpp" file of zeus, that gets $\mathbf{T_{oi}}$. It is then inverted can can be used in Equation 10. This transformation matrix will change with each change in position of the car in motion.

The more difficult transformation matrix to get is $\mathbf{T_{ci}}$. A series of transformation matrices can be computed using the "get_transform" function in the "transform_utils.cpp" file. The $\mathbf{T_{ci}}$ can be broken down as shown in Equation 11.

$$\mathbf{T_{ci}} = \mathbf{T_{cv}}\mathbf{T_{vi}} \tag{12}$$

Both $\mathbf{T_{cv}}$ and $\mathbf{T_{vi}}$ are directly extracted with the "get_transform" function. $\mathbf{T_{ci}}$ will remain constant for all the samples collected from the rosbag as it translates between locations stationary relative to the motion of the car.

Now the function for transforming the points is complete and the result of Equation 10 is used in Equations 6 and 7. The final step is finding the form of the Jacobian from Equations 6 and 7 to be used Equation 3.

[2]The disparity form is a 4 x 1 vector form that is normalized by the 3rd element, or z coordinate of the vector $\mathbf{x_i'} = \begin{bmatrix} u & v & 1 & d \end{bmatrix}^T$

2.3 The Jacobian

The last portion of this problem is computing the Jacobian, $\mathbf{J}$, of $\mathbf{f}(\mathbf{x_i}; \mathbf{p})$. $\mathbf{J}$ here will be a 2 x 4 matrix with the following form

$$J = \frac{\partial(u_i, v_i)}{\partial(X, Y, Z, W)} = \begin{bmatrix} \frac{\partial u_i}{\partial X} & \frac{\partial u_i}{\partial Y} & \frac{\partial u_i}{\partial Z} & \frac{\partial u_i}{\partial W} \\ \frac{\partial v_i}{\partial X} & \frac{\partial v_i}{\partial Y} & \frac{\partial v_i}{\partial Z} & \frac{\partial v_i}{\partial W} \end{bmatrix} \tag{13}$$

If J_i is taken with respect to $\mathbf{p}$ where $\mathbf{p} = (X, Y, W, 1)$, or the homogeneous form, the result is

$$J_i = \begin{bmatrix} \frac{P_D^{(i)} p_{00}^{(i)} - P_{N1}^{(i)} p_{20}^{(i)}}{P_D^{(i)2}} & \frac{P_D^{(i)} p_{01}^{(i)} - P_{N1}^{(i)} p_{21}^{(i)}}{P_D^{(i)2}} & \frac{P_D^{(i)} p_{02}^{(i)} - P_{N1}^{(i)} p_{22}^{(i)}}{P_D^{(i)2}} & \frac{P_D^{(i)} p_{03}^{(i)} - P_{N1}^{(i)} p_{23}^{(i)}}{P_D^{(i)2}} \\ \frac{P_D^{(i)} p_{10}^{(i)} - P_{N2}^{(i)} p_{20}^{(i)}}{P_D^{(i)2}} & \frac{P_D^{(i)} p_{11}^{(i)} - P_{N2}^{(i)} p_{21}^{(i)}}{P_D^{(i)2}} & \frac{P_D^{(i)} p_{12}^{(i)} - P_{N2}^{(i)} p_{22}^{(i)}}{P_D^{(i)2}} & \frac{P_D^{(i)} p_{13}^{(i)} - P_{N2}^{(i)} p_{23}^{(i)}}{P_D^{(i)2}} \end{bmatrix} \tag{14}$$

where

$$P_D^{(i)} = p_{20}^{(i)} X + p_{21}^{(i)} Y + p_{22}^{(i)} Z + p_{23}^{(i)} \tag{15}$$

$$P_{N1}^{(i)} = p_{00}^{(i)} X + p_{01}^{(i)} Y + p_{02}^{(i)} Z + p_{03}^{(i)} \tag{16}$$

$$P_{N2}^{(i)} = p_{10}^{(i)} X + p_{11}^{(i)} Y + p_{12}^{(i)} Z + p_{13}^{(i)} \tag{17}$$

This was the final step needed to start using the algorithm.

3 3D Rays Approach

Alternatively to the previously outlined approach, the 3D rays approach that only goes through the samples once.

3.1 Constructing Rays

The first important component for constructing the rays is the camera centres, $\mathbf{c_j}$, for the j samples that are used in the algorithm. $\mathbf{c_j}$ are the 3D world location of the camera centre, accounting for the focal length of the camera. It can usually be approximated as the location of the camera in the odometry frame. These points are the origins of the rays. The second component is the unit vector, $\hat{\mathbf{v}}_\mathbf{j}$, that points from the camera centre to the point on the camera image where the object centre is labelled, called $\mathbf{x_j}$. $\mathbf{x_j}$ are in pixel coordinates and the homogeneous form meaning a 1 is appended to the end of the vector, making it 3x1. The equation for the $\hat{\mathbf{v}}_\mathbf{j}$ is

$$\hat{\mathbf{v}}_\mathbf{j} = \frac{\mathbf{E^{-1}K^{-1}x_j}}{\|\mathbf{E^{-1}K^{-1}x_j}\|} \tag{18}$$

which is just the unit norm of the the numerator of this equation. As outlined in section 2, $\mathbf{E^{-1}}$ is the inverse of transformation matrix from the odometry frame to the camera frame but in its 3x4 form meaning the last row is dropped from

its previous inverse form. Similarly, $\mathbf{K^{-1}}$ is the inverse of the camera intrinsic matrix in it's 4x3 form meaning the last column of the 4x4 inverse is dropped. The multiplication of these two matrices is essentially the inverse of the 3x3 form of the camera matrix.

The goal of this approach is the find the point on the each of the rays, denoted $\mathbf{q_j}$, associated with each sample image that has the minimum distance to the ground truth location of the object's centre. The general equation for this distance is

$$\|\mathbf{c_j} + \hat{\mathbf{v}}_\mathbf{j}\mathbf{d_j} - \mathbf{p}\|^2 \tag{19}$$

and

$$d_j = \hat{\mathbf{v}}_\mathbf{j} \cdot (\mathbf{p} - \mathbf{c_j}) \tag{20}$$

where d_j is the minimum distance between the ground truth point $\mathbf{p}$ and the 3D rays associated with the sample images, making $\mathbf{q_j}$

$$\mathbf{q_j} = \mathbf{c_j} + (\hat{\mathbf{v}}_\mathbf{j}\hat{\mathbf{v}}_\mathbf{j}^\mathbf{T})(\mathbf{p} - \mathbf{c_j}) \tag{21}$$

Then the squared distance between these two points are

$$r_j^2 = \|(\mathbf{I} - \hat{\mathbf{v}}_\mathbf{j}\hat{\mathbf{v}}_\mathbf{j}^\mathbf{T})(\mathbf{p} - \mathbf{c_j}\|^2 \tag{22}$$

Using this as a linear least squares problem and summing over all the sample images, we can solve for $\mathbf{p}$

$$\mathbf{p} = \left[\sum_j (\mathbf{I} - \hat{\mathbf{v}}_\mathbf{j}\hat{\mathbf{v}}_\mathbf{j}^\mathbf{T})\right]^{-1} \left[\sum_j (\mathbf{I} - \hat{\mathbf{v}}_\mathbf{j}\hat{\mathbf{v}}_\mathbf{j}^\mathbf{T})\right] \tag{23}$$

Therefore, a final value for $\mathbf{p}$ is produced that minimizes the distance between itself and the points that lie closest to it on the 3D rays.

4 Comparing the Two Approaches

Both approaches are acceptable ways to approach the triangulation problem but each have different advantages and disadvantages. The iterative non-linear least squares approach is more statistically optimal and works better with cameras with varying closeness to the 3D point of interest however it also requires an initial guess for the algorithm and it's range of convergence prevents the user from using any initial guess. This means that an additional algorithm, even the 3D rays approach, can be used to get that initial guess or some method of manually finding a good enough estimate for the location must be used.

The 3D rays approach doesn't require an initial guess but can perform worse than the previous approach as it doesn't have any method for improving upon the one estimate it gives. While this is certainly a concern, it can give very good estimates and can be computationally less expensive as it doesn't deal with multiple iterations.

Depending on the requirements for your application, select an algorithm that works best.

5 Conclusion

All of the algorithms and fundamentals of math for this document were pulled of Szeliski's *Computer Vision: Algorithms and Applications* referenced at after this section. For more information on other similar algorithms or computer vision fundamentals, it's highly recommended to skim through Chapter 2, 6, and 7 of the textbook. The mentioned algotrithms can be implemented into C++ or python use the Eigen and numpy libraries respectfully.