

# Pose Estimation From Known World Points

Clark N. Taylor  
Brigham Young University  
taylor@ee.byu.edu

January 23, 2008

## 1 Introduction

When using a calibrated camera, it is possible to estimate the location and attitude (angular orientation) of the camera when a picture of three or more points of known location is taken. Generally, if only three objects are in view, there is an exact answer for only those three points. However, the algorithm to compute the pose (location and attitude) of the camera is numerically unstable, amplifying noise in the image location or real-world location of the points. Therefore, more than 3 points is typically used.

While several algorithms exist for computing the pose from the real-world and image locations of 3 or more objects, these notes focus specifically on the method in [1]. Note that some example MATLAB code implementing this algorithm can be found on the website with these notes.

Before we continue with a description of the algorithm let me just make one note. This algorithm returns *the pose of the camera with respect to the objects being imaged*. Therefore, if the objects being imaged are on a moving platform (i.e. the ground robot), the only information that can be obtained from this approach is the pose of the camera **in relation to the ground robot**. If you are observing fixed objects in your image, on the other hand, then the pose returned is in relation to the coordinate frame of those objects.

In the sections that follow, I first introduce some fundamental concepts that must be understood before the actual algorithm can be discussed. Then in section 3, I introduce the algorithm itself.

## 2 Fundamentals

### 2.1 The Projection Operator

Assume we have a vector  $\mathbf{v}$  and a point  $\mathbf{p}$ . We would like to “project”  $\mathbf{p}$  onto  $\mathbf{v}$ . In other words, what is the closest point on the ray formed by  $\mathbf{v}$  to  $\mathbf{p}$ . Assuming both  $\mathbf{v}$  and  $\mathbf{p}$  are column vectors, then the dot-product of the two is  $\mathbf{v}^T \mathbf{p}$ . The dot-product gives the length along  $\mathbf{v}$  that I should move to find the closest point to  $\mathbf{p}$ , times the length of  $\mathbf{v}$ . To find the closest point on  $\mathbf{v}$  to  $\mathbf{p}$ , therefore, I should normalize  $\mathbf{v}$ . Therefore, the projection operator for vector  $\mathbf{v}$  is:

$$\frac{\mathbf{v}\mathbf{v}^T}{\mathbf{v}^T \mathbf{v}} \quad (2.1)$$

Multiplying this matrix times any point  $\mathbf{p}$  will give the closest point on  $\mathbf{v}$  to  $\mathbf{p}$ .

## 2.2 SVD (The Singular Value Decomposition)

One of the most basic operations in linear algebra is the SVD. Any matrix ( $\mathbf{A}$ ) of size  $m \times n$  can be represented as:

$$\mathbf{A}_{m \times n} = \mathbf{U}_{m \times m} \mathbf{\Sigma}_{m \times n} \mathbf{V}_{n \times n}^T \quad (2.2)$$

where  $\mathbf{\Sigma}$  is a diagonal matrix and both  $\mathbf{U}$  and  $\mathbf{V}$  are orthogonal matrices (i.e.  $\mathbf{U}\mathbf{U}^T = \mathbf{I}$ ). In addition, the entries of  $\mathbf{\Sigma}$  are placed on the diagonal in descending order. We will not focus on the SVD here, but suffice it to say that there are many libraries that exist for computing the SVD.

## 2.3 The Best Coordinate Frame Transform Between Two Sets of Points

Assume you have two sets of points  $\mathbf{p}$  and  $\mathbf{q}$ . In addition, you know the one-to-one relationship between the points in each set. How could I determine the rotation and translation that best maps  $\mathbf{p}$  into  $\mathbf{q}$ . The following algorithm can be used (this is based on the solution to the *Orthogonal Procrustes Problem*)

1. Find the mean of both  $\mathbf{p}$  and  $\mathbf{q}$  ( $\bar{\mathbf{p}}$  and  $\bar{\mathbf{q}}$ ). Subtract the mean from all the points.
2. Form a matrix by summing the outer products of the normalized points.  $(\sum_{i=1}^N \mathbf{p}_i \mathbf{q}_i^T)$ .
3. Take the SVD of this matrix, yielding  $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ .
4. The optimal rotation between the points is  $\mathbf{R} = \mathbf{V}\mathbf{U}^T$ .
5. The optimal translation between the points is  $\mathbf{t} = \bar{\mathbf{q}} - \mathbf{R}\bar{\mathbf{p}}$

The “optimal” rotation and translation here are the rotation and translation that minimize the L2 norm of the distances between  $\mathbf{R}\mathbf{p} + \mathbf{t}$  and  $\mathbf{q}$ .

## 3 The Algorithm

With the fundamentals above, we can now estimate the best rotation and translation that yields image points  $\mathbf{v}$  from world points  $\mathbf{p}$ . We cannot directly utilize the solution to the Orthogonal Procrustes Problem (termed the *Optimal Absolute Orientation Solution* in [1]) because we are not mapping a set of world points onto a set of world points. Rather, we are taking a set of world points and mapping them onto projected points in the image plane. Therefore, we must modify the Optimal Absolute Orientation Solution.

Assume that  $\mathbf{R}$  and  $\mathbf{t}$  are known between the objects being viewed ( $\mathbf{p}$ ) and the camera, and no noise is present in any observations. In this case, we can first move the points  $\mathbf{p}$  into the camera’s coordinate frame using  $\mathbf{R}\mathbf{p} + \mathbf{t}$ . When projecting  $\mathbf{p}$  onto the image location vectors  $\mathbf{v}$ , no movement would

occur because there is no noise. However, in the case where there is noise, we would like to find the  $\mathbf{R}$  and  $\mathbf{t}$  that minimize the difference between  $\mathbf{p}$  and its projections onto the vectors  $\mathbf{v}$  defined by the image locations observed. We define:

$$\mathbf{q} = \frac{\mathbf{v}\mathbf{v}^T}{\mathbf{v}^T\mathbf{v}}(\mathbf{R}\mathbf{p} + \mathbf{t}) \quad (3.1)$$

and find the optimal rotation and translation between  $\mathbf{p}$  and  $\mathbf{q}$  using the Optimal Absolute Orientation Solution. This new rotation and translation is used to find a new set of  $\mathbf{q}$ , and the algorithm is repeated until the estimates of  $\mathbf{R}$  and  $\mathbf{t}$  no longer change from iteration to iteration.

### 3.1 Implementation Issues

The algorithm above assumes that a  $\mathbf{R}$  and  $\mathbf{t}$  are known to begin with. In fact, any  $\mathbf{R}$  and  $\mathbf{t}$  can be chosen. However, if a  $\mathbf{R}$  and  $\mathbf{t}$  that are close to the true answer is chosen, (1) the algorithm is more likely to return the true pose and (2) the algorithm will usually take fewer iterations to converge, minimizing the computation time required. Eventually, you will probably want to get an initial pose estimate from the auto-pilot, and then iterate from that point.

Also note that all values used in this algorithm are in the world space. Therefore, pixel locations from the image must be run through the inverse calibration matrix and any radial distortion correction *before* applying this algorithm.

## References

- [1] “Fast and Globally Convergent Pose Estimation from Video Images”, by Chien-Ping Lu, Gregory D. Hager, and Eric Mjolsness, in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 22(6), June 2000