# $P2\Pi$: A Minimal Solution for Registration of 3D Points to 3D Planes

Srikumar Ramalingam, Yuichi Taguchi, Tim K. Marks, and Oncel Tuzel

Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA, USA

**Abstract.** This paper presents a class of minimal solutions for the 3D-to-3D registration problem in which the sensor data are 3D points and the corresponding object data are 3D planes. In order to compute the 6 degrees-of-freedom transformation between the sensor and the object, we need at least six points on three or more planes. We systematically investigate and develop pose estimation algorithms for several configurations, including all minimal configurations, that arise from the distribution of points on planes. The degenerate configurations are also identified. We point out that many existing and unsolved 2D-to-3D and 3D-to-3D pose estimation algorithms involving points, lines, and planes can be transformed into the problem of registering points to planes. In addition to simulations, we also demonstrate the algorithm's effectiveness in two real-world applications: registration of a robotic arm with an object using a contact sensor, and registration of 3D point clouds that were obtained using multi-view reconstruction of planar city models.

## 1 Introduction and previous work

The problem of 3D-to-3D registration is one of the oldest and most fundamental problem in computer vision, photogrammetry, and robotics, with numerous application arease including object recognition, tracking, localization and mapping, augmented reality, and medical image alignment. Recent progress in the availability of 3D sensors at reasonable cost have further accelerated the need for such problems. The registration problem can generally be seen as two subproblems: a correspondence problem, and a problem of pose estimation given the correspondence. Both of these problems are intertwined, and the solution of one depends on the other. This paper addresses the solution to both problems, although the major emphasis is on the second one.

Several 3D-to-3D registration scenarios are possible depending on the representation of the two 3D datasets: 3D points to 3D points, 3D lines to 3D planes, 3D points to 3D planes, etc. [1]. For the registration of 3D points to 3D points, iterative closest point (ICP) and its variants have been the gold standard in the last two decades [2, 3]. These algorithms perform very well with a good initialization. Hence for the case of 3D points to 3D points, the main unsolved problem is the initial coarse registration.

The registration of 3D lines to 3D planes and the registration of 3D points *with normals* to 3D planes were considered in [4, 5]. (In this paper, we register 3D points without normals to 3D planes.) Recently, there have been several registration algorithms that focus on solving both the correspondence and pose estimation [4, 6, 7], primarily by casting the correspondence problem as a graph theoretical one. The correspondence

problem maps to a class of NP-hard problems such as minimum vertex cover [8] and maximum clique [9]. In this paper, we address the correspondence problem by formulating it as a maximum clique problem.

The main focus of this paper is on solving for the point-to-plane registration given the correspondence. Despite several existing results in 3D-to-3D registration problems, the registration of points to planes has received very little attention. However, in practice many registration problems can be efficiently solved by formulating them as point-to-plane. Iterative approaches exist for this problem [10, 1]. In [1], the authors specifically mention that their algorithms had difficulties with point-to-plane registration and pointed out the need for a minimal solution. The minimal solution developed here provides a clear understanding of degenerate cases of the point-to-plane registration.

The development of minimal solutions in general has been beneficial in several vision problems [11–16]. Minimal solutions have proven to be less noise-prone than non-minimal algorithms, and they have been quite useful in practice as hypothesis generators in hypothesize-and-test algorithms such as RANSAC [17]. Our minimal solution for the point-to-plane registration problem also comes with an additional advantage: it dramatically reduces the search space in the correspondence problem.

To validate our theory we show an exhaustive set of simulations and two compelling real-world proof-of-concept experiments: registration of a robotic arm with an object using contact sensor, and registration of 3D point clouds obtained using multi-view reconstruction on 3D planar city models.

*Problem statement:* Our main goal is to compute the pose (3D translation and 3D rotation) of a sensor with respect to an object (or objects) for which a 3D model consisting of a set of planes is already known. The sensor provides the 3D coordinates of a small set of points on the object, measured in the sensor coordinate frame. We are given $N$ points $P_1^0, P_2^0, P_3^0, ..., P_N^0$ from the sensor data and $M$ planes $\Pi_1^0, \Pi_2^0, \Pi_3^0, ..., \Pi_M^0$ from the 3D object. We subdivide the original problem into two sub-problems:

- Compute the correspondences between the 3D points in the sensor data and the planes in the 3D object.
- Given these correspondences, compute the rotation and translation $(\mathsf{R}_{s2w}, \mathbf{T}_{s2w})$ between the sensor and the object. We assume that the object lies in the world reference frame, as shown in Figure 1.

In this paper, we explain our solution to the second problem (pose estimation given the correspondences) in Section 2 before discussing the correspondence problem in Section 3.

## 2 Pose estimation

In this section, we develop the algorithms for pose estimation given the correspondences between the 3D points and their corresponding planes. Here we assume that the correspondences are already known—a method for computing the correspondences is explained later, in Section 3. We systematically consider several cases in which we know the distribution of the points on the planes (how many points correspond to each plane), developing a customized pose estimation algorithm for each case. We denote

each configuration as *Points*$(a_1, a_2, ..., a_n) \leftrightarrow$ *Planes*$(n)$, where $n = \{3, 4, 5, 6\}$ is the number of distinct planes in which the points lie, and $a_i$ is the number of points that lie in the $i$th plane. The correspondence between a single point and a plane will yield a single coplanarity equation. Since there are 6 unknown degrees of freedom in $(\mathsf{R}_{s2w}, \mathbf{T}_{s2w})$, we need at least 6 point-to-plane correspondences to solve the pose estimation problem. There are also degenerate cases in which 6 correspondences are not sufficient. Although the individual algorithms for the various cases are slightly different, their underlying approach is the same. The algorithms for all cases are derived using the following three steps:

- *The choice of intermediate coordinate frames:* We transform the sensor and the object to intermediate coordinate frames to reduce the degree of the resulting polynomial equations. In addition, if the transformation results in a decrease in the number of degrees of freedom in the pose between the sensor and object, then the rotation $\mathsf{R}$ and the translation $\mathbf{T}$ are expressed using fewer variables.
- *The use of coplanarity constraints:* From the correspondences between the points and planes, we derive a set of coplanarity constraints. Using a linear system involving the derived coplanarity constraints, we express the unknown pose variables in a subspace spanned by one or more vectors.
- *The use of orthonormality constraints:* Finally, we use the appropriate number of orthonormality constraints from the rotation matrix to determine solutions in the subspace just described.

### 2.1 The choice of intermediate coordinate frames

As shown in Figure 1, we denote the original sensor frame (in which the points reside) and the world reference frame (where the planes reside) by $\mathcal{S}^0$ and $\mathcal{W}^0$, respectively. Our goal is to compute the transformation $(\mathsf{R}_{s2w}, \mathbf{T}_{s2w})$ that transforms the 3D points from the sensor frame $\mathcal{S}^0$ into the world reference frame $\mathcal{W}^0$. A straightforward application of coplanarity constraints in the case of 6 points would result in 6 linear equations involving 12 variables (the 9 elements of the rotation matrix $\mathsf{R}_{s2w}$ and the 3 elements of the translation vector $\mathbf{T}_{s2w}$). To solve for these variables, we would need at least 6 additional equations; these can be 6 quadratic orthonormality constraints. The solution of such a system may eventually result in a polynomial equation of degree $64 = 2^6$, which would have 64 solutions (upper bound as per Bezout's theorem), and the computation of such solutions would likely be infeasible for many applications.

To overcome this difficulty, we first transform the sensor and world reference frames $\mathcal{S}^0$ and $\mathcal{W}^0$ to two new intermediate coordinate frames, which we call $\mathcal{S}$ and $\mathcal{W}$. After this transformation, our goal is to find the remaining transformation $(\mathsf{R}, \mathbf{T})$ between the intermediate reference frames $\mathcal{S}$ and $\mathcal{W}$. We choose $\mathcal{S}$ and $\mathcal{W}$ so as to minimize the number of variables in $(\mathsf{R}, \mathbf{T})$ that we need to solve for. A similar idea has been used in other problem domains [18]. We now define the transformations from the initial reference frames to the intermediate frames and prove that these transformations are always possible using a constructive argument.

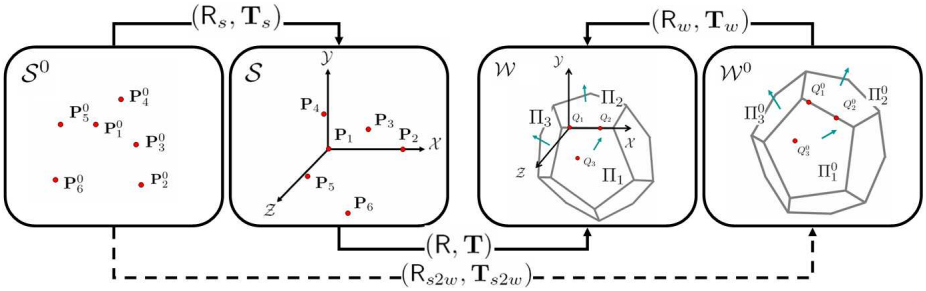



**Fig. 1.** The basic idea of coordinate transformation for pose estimation. It is always possible to transform the sensor coordinate system such that a chosen triplet of points $(P_1, P_2, P_3)$ lie respectively at the origin, on the $\mathcal{X}$ axis, and on the $\mathcal{XY}$ plane. On the other hand, the object coordinate frame can always be transformed such that $\Pi_1$ coincides with the $\mathcal{XY}$ plane and ($\Pi_2$) contains the $\mathcal{X}$ axis.

**Transformation from $\mathcal{S}^0$ to $\mathcal{S}$** As shown in Figure 1, we represent the $i$th point in $\mathcal{S}^0$ using the notation $P_i^0$ and the same point in $\mathcal{S}$ using $P_i$. We define the transformation $(\mathsf{R}_s, \mathbf{T}_s)$ as the one that results in the points $(P_1, P_2, P_3)$ satisfying the following conditions: (a) $P_1$ lies at the origin, (b) $P_2$ lies on the positive $\mathcal{X}$ axis, and (c) $P_3$ lies in the $\mathcal{XY}$ plane. Note that the points $P_i^0$ are already given in the problem statement, and the transformation to the points $P_i$ can be easily computed using the above conditions.

**Transformation from $\mathcal{W}^0$ to $\mathcal{W}$** We similarly represent the $i$th plane in $\mathcal{W}^0$ using the notation $\Pi_i^0$ and the same plane in $\mathcal{W}$ using $\Pi_i$. We define the transformation as the one that results in the planes $\Pi_i$ satisfying the following two conditions: (a) $\Pi_1$ coincides with the $\mathcal{XY}$ plane, and (b) $\Pi_2$ contains the $\mathcal{X}$ axis.

Assume that $Q_1^0$ and $Q_2^0$ are two points on the line of intersection of the two planes $\Pi_1^0$ and $\Pi_2^0$. Let $Q_3^0$ be any other point on the plane $\Pi_1^0$. Let $Q_1$, $Q_2$, and $Q_3$ denote the same 3D points after the transformation from $\mathcal{W}^0$ to $\mathcal{W}$. The required transformation $(\mathsf{R}_w, \mathbf{T}_w)$ is the one that maps the triplet $(Q_1^0, Q_2^0, Q_3^0)$ to $(Q_1, Q_2, Q_3)$. Note that three points $Q_i^0$ satisfying the description above can be easily determined from the planes $\Pi_i^0$, and the transformation from points $Q_i^0$ to points $Q_i$ can be computed in the same way as the transformation described above from points $P_i^0$ to points $P_i$.

We denote the 3D points after the transformation as follows:

$$P_1 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix},\ P_2 = \begin{pmatrix} X_2 \\ 0 \\ 0 \end{pmatrix},\ P_3 = \begin{pmatrix} X_3 \\ Y_3 \\ 0 \end{pmatrix},\ \text{and } P_i = \begin{pmatrix} X_i \\ Y_i \\ Z_i \end{pmatrix} \text{ for } i = \{4, 5, 6\}. \quad (1)$$

We write the equations of the planes after the transformation as follows:

$$Z = 0 \ :\ \Pi_1 \quad (2)$$

$$B_2 Y + C_2 Z = 0 \ :\ \Pi_2 \quad (3)$$

$$A_i X + B_i Y + C_i Z + D_i = 0 \ :\ \Pi_i, \text{ for } i = \{3, 4, 5, 6\} \quad (4)$$

**Point-to-plane assignment** Depending on the particular configuration *Points*$(a_1, ..., a_n)$ $\leftrightarrow$ *Planes*$(n)$ of the points and planes, we choose which sensor points correspond to each of $P_1, P_2, \ldots$, and which object planes correspond to each of $\Pi_1, \Pi_2, \ldots$, so as to minimize the number of variables in the transformation between the intermediate frames.

In the remainder of this subsection, and in the following subsections 2.2 and 2.3, we explain the method in the context of a particular example: namely, the configuration *Points*$(3, 2, 1) \leftrightarrow$ *Planes*$(3)$. For this configuration, we may without loss of generality assume the following correspondences between the points and the planes:

$$\Pi_1 \Longleftarrow \{P_1, P_2, P_3\}, \quad \Pi_2 \Longleftarrow \{P_4, P_5\}, \quad \Pi_3 \Longleftarrow \{P_6\}. \tag{5}$$

As a result of this assignment, the plane corresponding to the three points $\{P_1, P_2, P_3\}$ and the plane $\Pi_1$ are both mapped to the $\mathcal{XY}$ plane. The final rotation $(\mathsf{R})$ and translation $(\mathbf{T})$ between the intermediate sensor coordinate frame $\mathcal{S}$ and the intermediate object coordinate frame $\mathcal{W}$ must preserve the coplanarity of these three points and their corresponding plane. Thus, the final transformation can be chosen so as to map all points on the $\mathcal{XY}$ plane to points on the $\mathcal{XY}$ plane. In other words, the rotation should be only along the $\mathcal{Z}$ axis and the translation along the $\mathcal{X}$ and the $\mathcal{Y}$ axes. There are two pairs of rotation and translation that satisfy this constraint:

$$\mathsf{R}_1 = \begin{pmatrix} R_{11} & R_{12} & 0 \\ -R_{12} & R_{11} & 0 \\ 0 & 0 & 1 \end{pmatrix}, \mathbf{T}_1 = \begin{pmatrix} T_1 \\ T_2 \\ 0 \end{pmatrix}; \quad \mathsf{R}_2 = \begin{pmatrix} R_{11} & R_{12} & 0 \\ R_{12} & -R_{11} & 0 \\ 0 & 0 & -1 \end{pmatrix}, \mathbf{T}_2 = \begin{pmatrix} T_1 \\ T_2 \\ 0 \end{pmatrix} \tag{6}$$

By choosing assignment (5) and separately formulating $\mathsf{R}_1$ and $\mathsf{R}_2$, we have minimized the number of degrees of freedom to solve for in the transformation between the intermediate frames of reference. Note that $\mathsf{R}_1$ and $\mathsf{R}_2$ are related to each other by a $180^\circ$ rotation about the $\mathcal{X}$ axis. Below, we explain the algorithm for solving for $\mathsf{R}_1$ and $\mathbf{T}_1$.

### 2.2 The use of coplanarity constraints

To explain our method's use of coplanarity constraints (and orthonormality constraints), we continue with the example of the specific configuration *Points*$(3, 2, 1) \leftrightarrow$ *Planes*$(3)$. We know that the points $P_4$ and $P_5$ lie on the plane $\Pi_2$, whose equation is given by (3). This implies that these points must satisfy the following coplanarity constraints:

$$B_2(-R_{12}X_i + R_{11}Y_i + T_2) + C_2Z_i = 0, \text{ for } i = \{4, 5\} \tag{7}$$

Similarly, the constraint from the third plane $\Pi_3$ is given below:

$$A_3(R_{11}X_6 + R_{12}Y_6 + T_1) + B_3(-R_{12}X_6 + R_{11}Y_6 + T_2) + C_3Z_6 + D_3 = 0 \tag{8}$$

Using the coplanarity constraints (7), (8), we construct the following linear system:

$$\underbrace{\begin{pmatrix} B_2Y_4 & -B_2X_4 & 0 & B_2 \\ B_2Y_5 & -B_2X_5 & 0 & B_2 \\ A_3X_6 + B_3Y_6 & A_3Y_6 - B_3X_6 & A_3 & B_3 \end{pmatrix}}_{\mathcal{A}} \begin{pmatrix} R_{11} \\ R_{12} \\ T_1 \\ T_2 \end{pmatrix} = \begin{pmatrix} -C_2Z_4 \\ -C_2Z_5 \\ -C_3Z_6 - D_3 \end{pmatrix} \tag{9}$$

The matrix $\mathcal{A}$ consists of known values and has rank 3. As there are 4 variables in the linear system, we can obtain their solution in a subspace spanned by one vector:

$$\begin{pmatrix} R_{11} & R_{12} & T_1 & T_2 \end{pmatrix}^T = \begin{pmatrix} u_1 & u_2 & u_3 & u_4 \end{pmatrix}^T + l_1 \begin{pmatrix} v_1 & v_2 & v_3 & v_4 \end{pmatrix}^T, \tag{10}$$

where the values $u_i$, $v_i$ are known, and $l_1$ is the only unknown variable.

### 2.3 The use of orthonormality constraints

We can solve for the unknown variable $l_1$ using a single orthonormality constraint ($R_{11}^2 + R_{12}^2 = 1$) for the rotation variables.

$$(u_1 + l_1 v_1)^2 + (u_2 + l_1 v_2)^2 = 1 \tag{11}$$

By solving the above equation, we obtain two different solutions for $l_1$. As a result, we obtain two solutions for the transformation $(\mathsf{R}_1, \mathbf{T}_1)$. Since we can similarly compute two solutions for $(\mathsf{R}_2, \mathbf{T}_2)$, we finally have four solutions for $(\mathsf{R}, \mathbf{T})$. Using the obtained solutions for $(\mathsf{R}, \mathbf{T})$, the transformation between the original coordinate frames $(\mathsf{R}_{s2w}, \mathbf{T}_{s2w})$ can be easily computed.

**Visualization of the four solutions:** There is a geometric relationship between the multiple solutions obtained for the transformation $(\mathsf{R}, \mathbf{T})$. For example, in Figure 2(a), we show the four solutions derived above, for a special case in which the 3 planes are orthogonal to each other. All of the solutions satisfy the same set of plane equations, but they exist in different octants. Every solution is just a rotation of another solution about one of the three axes by $180°$. If we slightly modify the planes so that they are no longer orthogonal, the different solutions start to drift away from each other.

### 2.4 Other variants

The example shown above is one of the easiest point-to-plane registration algorithms to derive. Several harder configurations also arise from the distribution of 6 (or more) distinct points on 3 or more planes (see Table 1). We have solved every case using the same intermediate transformation technique described above. All of the different scenarios, the corresponding assignments of points and planes, and the number of solutions are summarized in Table 1.

The key to solving each configuration is to determine a point-to-plane assignment that minimizes the number of variables appearing in the transformation $(\mathsf{R}, \mathbf{T})$ between the intermediate frames. In general, such an optimal assignment can be found by considering different point-to-plane assignments and checking the resulting coplanarity constraint equations for the 6 points and their corresponding planes. For example, in the configuration *Points*$(3, 2, 1) \leftrightarrow$ *Planes*$(3)$, the point-to-plane assignments given in (5) minimize the number of unknowns in the equations (6) for $(\mathsf{R}, \mathbf{T})$. Please see the Supplementary Materials for details of various configurations summarized in Table 1.

**Table 1.** Point-to-plane configurations and their solutions.

Each row of the table presents a different configuration, in which $n$ denotes the number of distinct planes and each $a_i$ refers to the number of points that lie in the $i$th plane. The first two rows show the degenerate cases for which there is an insufficient number of points or planes. The next four rows consider non-minimal solutions using more than 6 points. The remaining rows show several minimal configurations (each using exactly 6 points). The number of solutions is given, followed by the average number of real (non-imaginary) solutions in parentheses based on 1000 computations from the simulation described in Section 5. Processing time was measured using a MATLAB implementation on a 2.66 GHz PC; the symbol † indicates the use of Groebner basis methods [19]. The Supplementary Materials explain the derivations of the various configurations.

| $n$ | $(a_1, \ldots, a_n)$ | Assignment | # of Solutions | Process time (msec) |
|---|---|---|---|---|
| $< 3$ | – | – | degenerate | – |
| $n$ | $\sum a_i < 6$ | – | degenerate | – |
| 3 | (3,3,3) | $\Pi_1 \Leftarrow \{P_1, P_2, P_3\}, \Pi_2 \Leftarrow \{P_4, P_5, P_6\}, \Pi_3 \Leftarrow \{P_7, P_8, P_9\}$ | 2 (2) | 5 |
| 3 | (3,3,2) | $\Pi_1 \Leftarrow \{P_1, P_2, P_3\}, \Pi_2 \Leftarrow \{P_4, P_5, P_6\}, \Pi_3 \Leftarrow \{P_7, P_8\}$ | 2 (2) | 5 |
| 3 | (3,3,1) | $\Pi_1 \Leftarrow \{P_1, P_2, P_3\}, \Pi_2 \Leftarrow \{P_4, P_5, P_6\}, \Pi_3 \Leftarrow \{P_7\}$ | 2 (2) | 5 |
| 3 | (3,2,2) | $\Pi_1 \Leftarrow \{P_1, P_2, P_3\}, \Pi_2 \Leftarrow \{P_4, P_5\}, \Pi_3 \Leftarrow \{P_6, P_7\}$ | 2 (2) | 5 |
| 3 | (4,1,1) | – | degenerate | – |
| 3 | (3,2,1) | $\Pi_1 \Leftarrow \{P_1, P_2, P_3\}, \Pi_2 \Leftarrow \{P_4, P_5\}, \Pi_3 \Leftarrow \{P_6\}$ | 4 (4) | 6 |
| 3 | (2,2,2) | $\Pi_1 \Leftarrow \{P_5, P_6\}, \Pi_2 \Leftarrow \{P_3, P_4\}, \Pi_3 \Leftarrow \{P_1, P_2\}$ | 8 (4.4) | 140† |
| 4 | (3,1,1,1) | $\Pi_1 \Leftarrow \{P_1, P_2, P_3\}, \Pi_2 \Leftarrow \{P_4\}, \Pi_3 \Leftarrow \{P_5\}, \Pi_3 \Leftarrow \{P_6\}$ | 4 (2.8) | 6 |
| 4 | (2,2,1,1) | $\Pi_1 \Leftarrow \{P_5, P_6\}, \Pi_2 \Leftarrow \{P_3, P_4\}, \Pi_3 \Leftarrow \{P_2\}, \Pi_4 \Leftarrow \{P_1\}$ | 8 (3.6) | 140† |
| 5 | (2,1,1,1,1) | $\Pi_1 \Leftarrow \{P_5, P_6\}, \Pi_i \Leftarrow \{P_{6-i}\}, i = \{3, 4, 5\}$ | 16 (5.8) | 410† |
| 6 | (1,1,1,1,1,1) | $\Pi_i \Leftarrow \{P_{6-i+1}\}, i = \{1, 2, 3, 4, 5, 6\}$ | 16 (5.8) | 1200† |

**Special cases** If the points lie on the boundaries of the planes (i.e., every point lies on two planes), then 3 points are sufficient to compute the pose. A careful analysis shows that this problem is nothing but a generalized 3-point pose estimation problem [20].

**Degenerate cases** Table 1 includes several degenerate cases based on the number of points and planes. In addition, degeneracies can occur based on the geometry of the planes. In the case of 3 planes, if the $3 \times 3$ matrix consisting of all three normals has rank less than 3 (e.g., if two of the three planes are parallel), it is a degenerate configuration.

## 3 The correspondence problem

In the previous section, we assumed that the point-to-plane correspondences were known. In this section, we briefly describe a method to compute these correspondences. The basic idea of the correspondence problem and the geometrical constraints involved in identifying feasible correspondences are explained in detail in [5] using an interpretation tree approach. The same problem can also be formulated as graph-theoretical problems such as independent set, vertex cover and maximum clique [5, 8, 9].

Our goal in this section is to compute all of the feasible mappings (possible assignments) between the 3D points in the sensor domain and planes in the object. Feasible

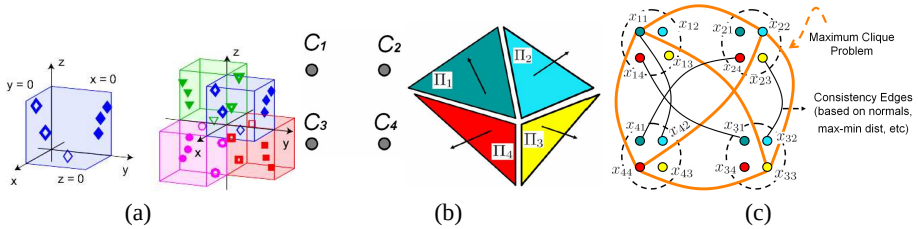



**Fig. 2.** (a) *Right* Visualization of 4 solutions for the points lying on 3 orthogonal planes. *Left:* Correct solution. (b) The problem of finding correspondences between clusters of points $C_i$ and planes $\Pi_j$. (c) This can be formulated as a maximum clique problem. Each node $x_{ij}$ in this graph represents a mapping between cluster $C_i$ and plane $\Pi_j$. An edge between two nodes is a consistency edge, signifying that both of these mappings can occur simultaneously without conflicting with the three constraints given in [5].

mappings refer to correspondences that satisfy the many geometrical constraints arising from the angles between the normals, pairwise distances, etc. [5]. Although such constraints do not always guarantee the correctness of the mappings, a wrong correspondence seldom exists satisfying all the constraints. In addition, since we use them in hypothesize-and-test algorithms such as RANSAC, outliers can be detected and removed.

In what follows, we briefly explain our approach using the maximum clique problem formulation. First, we cluster the points from the sensor into several planes, denoting the $i$th cluster as $C_i$. Note that each cluster may contain multiple points or even just a single point. As shown in Figure 2(b), our goal is to map these clusters to the corresponding planes $\Pi_j$ in the object. In order to do this, we construct a graph as shown in Figure 2(c). Every node in this graph $x_{ij}$ represents a mapping between the cluster $C_i$ (from the sensor) and the plane $\Pi_j$ (from the object). An edge between $x_{ij}$ and $x_{kl}$ is referred to as a consistency edge that signifies that both these mappings can occur simultaneously without conflicting with the three constraints given in [5]. The feasible correspondences between points and planes can be obtained by finding the maximum clique in the graph. A maximum clique for a graph refers to the largest subset of nodes in which each pair of nodes in the subset is connected by an edge. In the graph we constructed, finding a maximum clique provides us a set of mappings in which all possible pairwise consistencies are satisfied.

Several techniques can be used to solve these NP-hard problems [8, 7]. Since we use minimal approaches for our applications, we are not interested in the correspondences for all of the points in the registration problem. Instead, we are concerned with identifying a small number of point-to-plane correspondences (sufficient to resolve issues from degeneracies and outliers). In fact, one of the main advantages of the proposed minimal solution is that it only requires correspondences for a small number of points. This enabled us to use a simple tree-based search for finding the maximum cliques in the real-world experiments described in Section 5.

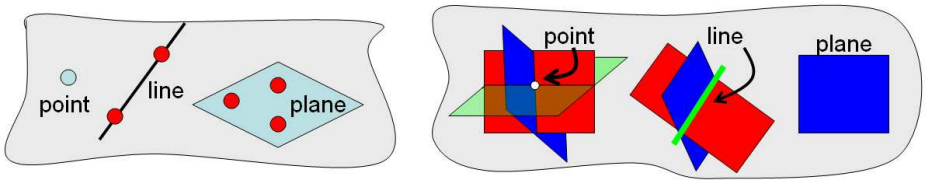



**Fig. 3.** A general framework to transform a given registration problem to a point-to-plane problem. *Left:* In the sensor data, we transform all geometrical entities (points, lines and planes) to points. A point is preserved as a point. In the case of lines and planes we sample two and three arbitrary points, respectively. *Right:* In the object data, we convert all geometrical entities to planes. A plane is preserved as a plane. Points and lines are parameterized using 3-plane and 2-plane representations, as shown.

## 4 A General Framework for Pose Estimation

We briefly sketch a unified pose estimation framework for most 2D-to-3D and 3D-to-3D registrations by first transforming the given problem to a point-to-plane registration problem. Several 2D-to-3D pose estimation algorithms have been proposed in the literature [6, 18, 10, 1, 21, 5, 4, 20]. All of these pose estimation algorithms involve the registration of one set of geometrical entities (points, lines, or planes) to another. For example, in the case of generalized pose estimation, we register three 3D points to the corresponding non-parametric projection rays from the cameras to compute the pose of the object with respect to the camera [20]. In the case of 2D-to-3D pose estimation using three lines, we can look at this problem as a registration of three interpretation planes (each formed by two projection rays corresponding to a single line) on three lines [18]. In the case of 3D-to-3D line-to-plane registration, we register lines from the sensor data to planes from the object [4]. In the case of 3D-to-3D point-to-point registration, we register points from sensor data to points in the object [6]. One could also propose registration algorithm involving mixture of geometrical entities and thereby we could have more than 20 2D-to-3D and 3D-to-3D registration scenarios. We emphasis that any of these pose estimation algorithms involving any combination of geometrical entities to any other combination could be transformed to a point-to-plane registration algorithm and solved using the following simple algorithm.

1. In the sensor data, we transform all the geometrical entities (points, lines and planes) to points. This is done using 2-point and 3-point representation of lines and planes respectively as shown in Figure 3.
2. In the object data, we transform all the geometrical entities to planes. This is done by 3-plane and 2-plane representations for points and lines, respectively. Note that the 3 planes passing through a point need not be orthogonal. Similarly, we use 2 non-orthogonal planes to represent a line. The appropriate choice of these planes plays a crucial role in obtaining an efficient pose estimation algorithm.
3. After these transformations, we can use our point-to-plane registration algorithm.

Details of the proposed generalized framework are given in the Supplementary Materials with examples on several registration problems.

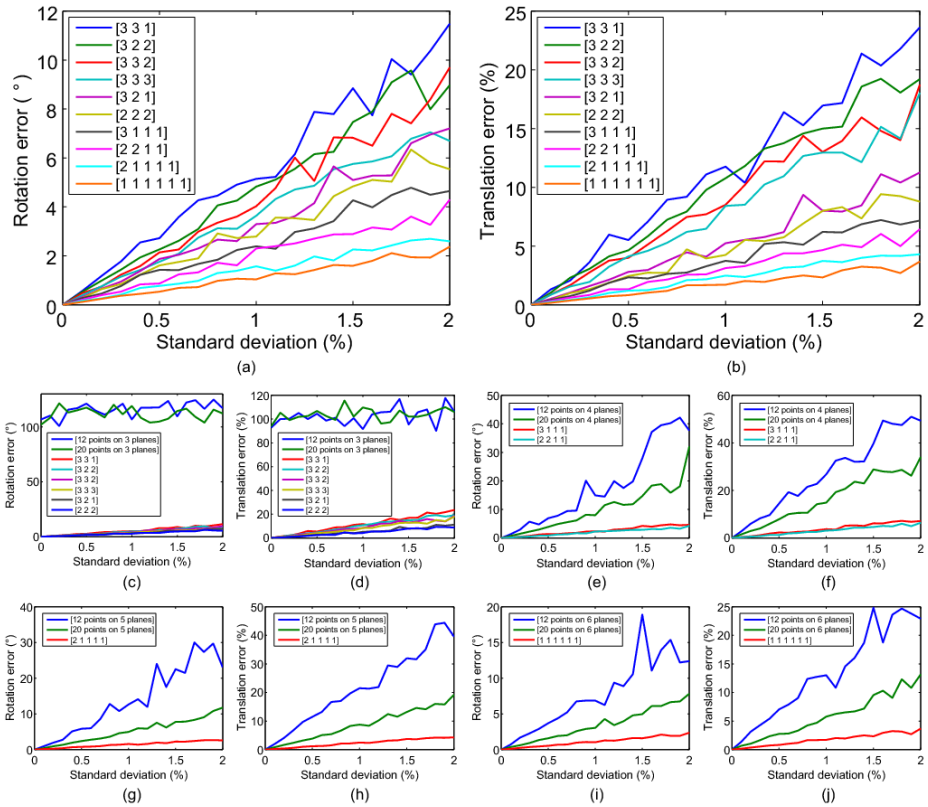
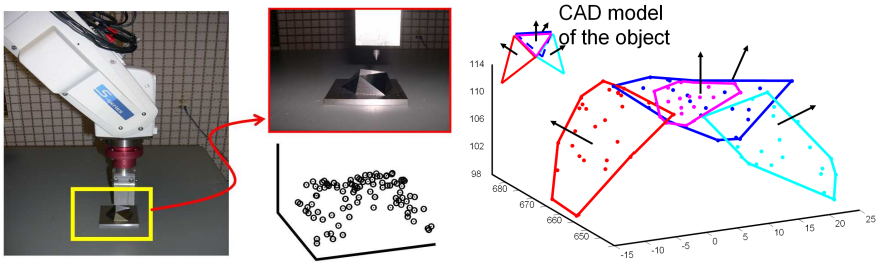



**Fig. 4.** Rotation and translation error for simulation data as a function of the level of noise in the test set. The noise standard deviation is expressed as a percentage of the size of the object. The legends list the configurations in order of decreasing error. (a,b) Results from our algorithm for all non-degenerate configurations shown in Table 1. Note that minimal solutions using 6 points provide lower errors than non-minimal solutions, and solutions for configurations with larger number of planes have lower errors. (b–j) Our minimal solutions compared to least square methods (using 12 and 20 points) for the same number of planes $n$: (c,d) $n = 3$, (e,f) $n = 4$, (g,h) $n = 5$, and (i,j) $n = 6$. Note that in the 3-plane case (b), least square methods completely fail due to rank degeneracy.

## 5 Experimental Results

**Simulations:** We analyzed the performance of our minimal solutions in simulations by generating 32 random planes inside a cube of side length 100 units. We randomly sampled 320 points on these planes within the cube. A test set was created by transforming all 320 points using a ground-truth rotation and translation, then adding Gaussian noise to each point.

We randomly selected $k$ points from the test set according to the point-to-plane configuration of the algorithm, then computed the rotation and translation using the points and the corresponding planes. The estimated transformation was then evaluated by us-



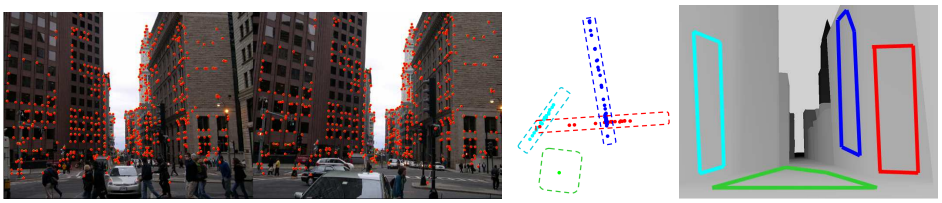

(a) (b)

**Fig. 5.** Real-world experiment with a 6-degrees-of-freedom robotic arm. (a) 3D contact position data were collected for 100 points on the surface using a built-in contact detection function and built-in encoders of the robotic arm. (b) Plane fitting of the 3D points and the correspondences of the points to the planes in the CAD model using the method of Section 3.

ing it to transform the other $320 - k$ points and computing the mean point-to-plane distance between the transformed points and their correct corresponding planes. Each trial consists of generating a test set, then repeating the selection of $k$ points and transformation estimation 100 times for this test set. Of the resulting 100 transformations, the solution for the trial is the one transformation that provides the minimum mean distance.

Figure 4 plots errors in estimated rotation and translation with varying noise levels. For each configuration, the errors plotted are the average of 100 trials. For each number of planes ($n = 3, 4, 5, 6$), we compare our minimal solutions for every possible configuration of 6 points (as well as the non-minimal configurations for 3 planes that were included in Table 1) to a least-squares solution for the same number of planes using 12 or 20 points without orthonormality constraints. In all cases, our minimal solutions yield smaller errors than the least squares method. Note that the least squares method completely fails in the case of three planes. Thus, our transformation is useful not only for the minimal configurations but also in non-minimal configurations such as $(3, 3, 3)$.

**Contact Sensor:** The first experiment, shown in Figure 5, was conducted using a 6-degree-of-freedom robotic arm with a built-in contact detection function. We used as the target object a partial surface of an icosahedron, of which four of the 20 faces are measurable, as shown in Figure 5. The robot automatically measured 100 points (contact positions) on the surface; each point was measured by first moving the probe to a random $x$, $y$ position and then moving down towards the surface (in the negative $z$ direction) until it sensed a contact. We clustered the points using a simple RANSAC-based plane fitting algorithm. There were four main clusters corresponding to the four planes of the icosahedron used in the experiment. Next, the method described in Section 3 was used to find the correspondences between these clusters and the planes in the 3D model. Given these correspondences, we applied our point-to-plane algorithm using several of the minimal 3-plane and 4-plane configurations. As in the simulations, we repeated the following process to determine the solution: randomly selecting $k$ points,



(a) (b) (c)

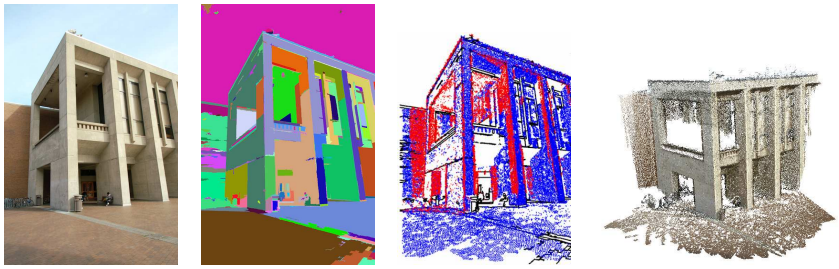
**Fig. 6.** (a) An input stereo pair of photos taken in Boston's financial district, overlaid with the points that we matched and reconstructed in 3D. (b) We identify four clusters in the reconstructed 3D points (a single point and three planar clouds of points) using a plane-fitting algorithm. (c) The four planes in the 3D city model corresponding to the identified clusters shown in (b).

solving for the transformation, and evaluating the mean distance of the transformed remaining points to the 3D model. The final point-to-plane distance error for all of the inliers was about $3\%$ of the overall size of the scene. The least squares method failed completely for the 3-plane case (similar to the results shown in Figure 4). In the 4-plane case, the least-squares error was about 10 times larger than the error of the minimal solutions.

**Registration of 3D point clouds to polyhedral architectural models:** Given a plane-approximated coarse 3D model of the city of Boston obtained from a commercial website (http://www.3dcadbrowser.com/), we performed localization within the map using a pair of images of a scene in Boston's financial district. To obtain 3D points from the image pair, we matched Harris features and applied standard structure-from-motion algorithms.

Using a RANSAC-based plane fitting algorithm, we fit planes to the reconstructed 3D points. We computed 3 planes from the reconstructed points as shown in Figure 6. A coarse initialization is manually provided and the nearest planes in the 3D model are identified. All of the planes shown in Figure 6(c) (more than 10 planes) were used from the 3D model of Boston. Using the method described in Section 3, we obtained the correspondences between four clusters (a single point and three planar clouds of points) and four planes in the 3D model. The plane corresponding to the ground had only one 3D point due to occlusion from pedestrians and cars. (Note that it was important to have at least one point on the ground in order to determine the vertical translation.) Applying our minimal algorithms for the 4-planes case yielded results with an error of just $0.05\%$ of the overall size of the scene.

Our point-to-plane registration algorithm can also be used for merging partial reconstructions obtained from multi-view reconstruction techniques [22, 23], as shown in Figure 7. In order to obtain a 3D model from 30 images, we subdivide the images into two clusters of 15 images each. We reconstruct 3D point clouds from each image cluster and use the superpixel segmentation of a common image to register them. The 3D points from the first cluster are reprojected onto the superpixel image and used to compute the plane parameters for each superpixel. (We eliminate superpixels with



(a) (b) (c) (d)

**Fig. 7.** Registering two point clouds, each generated by applying multi-view reconstruction techniques to 15 images. **(a)** One of the images used in 3D reconstruction. **(b)** superpixel segmentation of the image shown in (a). **(c)** The 3D points from the first (blue) and second (red) clouds are reprojected onto the superpixel image. The points from the first point cloud are used to compute the superpixel plane parameters, while the second point cloud is preserved as points. The correspondence between the points from the second cloud and the planes obtained from the first cloud are determined by the underlying superpixel. **(d)** 3D model after merging the two partial reconstructions from the two clusters. [**Best viewed in color**]

insufficient or non-planar points.) The superpixel segmentation of the common image gives us the correspondences between the points in the second cluster and the planes obtained from the first cluster. We obtain the 3D registration using a RANSAC framework, in which we select three or more non-degenerate planes (See section 2.4) and the corresponding minimum number of points.

Previous work merging partial 3D models obtained multi-view 3D reconstruction has used non-minimal iterative approaches [24]. However, initializing with a minimal solution, such as the one described here, may be critical for noisy 3D data. In addition, there are two general advantages of point-to-plane rather than point-to-point registration: (1) accuracy [25], (2) compact representation of the 3D models (about a million 3D points are represented using few hundred superpixel planes).

## 6 Discussion

The development of minimal algorithms for registering 3D points to 3D planes provides opportunities for efficient and robust algorithms with wide applicability in computer vision and robotics. Since 3D sensors typically do not perceive the boundaries of objects in the same way as 2D sensors, an algorithm that can work with points on the surfaces, rather than surface boundaries, is essential. In textureless 3D models, for example, it is easier to obtain point-to-plane correspondences than point-to-point and line-to-line correspondences.

**Acknowledgments:** We would like to thank Jay Thornton, Keisuke Kojima, John Barnwell, and Haruhisa Okuda for their valuable feedback, help and support.

## References


1. Olsson, C., Kahl, F., Oskarsson, M.: The registration problem revisited: Optimal solutions from points, lines and planes. In: CVPR. Volume 1. (2006) 1206–1213
2. Besl, P., McKay, N.: A method for registration of 3D shapes. In: PAMI. (1992)
3. Fitzgibbon, A.: Robust registration of 2d and 3d point sets. In Image and Vision Computing (2003)
4. Chen, H.: Pose determination from line-to-plane correspondences: Existence condition and closed-form solutions. PAMI **13** (1991) 530–541
5. Grimson, W., Lozano-Prez, T.: Model-based recognition and localization from sparse range or tactile data. MIT AI Lab, A.I. Memo 738 (1983)
6. Horn, B.: Closed-form solution of absolute orientation using unit quaternions. Journal of the Optical Society A **4** (1987) 629–642
7. Li, H., Hartley, R.: The 3D-3D registration problem revisited. In: ICCV. (2007) 1–8
8. Enqvist, O., Josephson, K., Kahl, F.: Optimal correspondences from pairwise constraints. In: ICCV. (2009)
9. Tu, P., Saxena, T., Hartley, R.: Recognizing objects using color-annotated adjacency graphs. In: In Lecture Notes in Computer Science: Shape, Contour and Grouping in Computer Vision. (1999)
10. Chen, Y., Medioni, G.: Object modeling by registration of multiple range images. In: ICRA. Volume 3. (1991) 2724–2729
11. Kukelova, Z., Pajdla, T.: A minimal solution to the autocalibration of radial distortion. In: CVPR. (2007)
12. Gao, X., Hou, X., Tang, J., Cheng, H.: Complete solution classification for the perspective-three-point problem. PAMI **25** (2003) 930–943
13. Stewenius, H., Nister, D., Kahl, F., Schaffalitzky, F.: A minimal solution for relative pose with unknown focal length. In: CVPR. (2005)
14. Stewenius, H., Nister, D., Oskarsson, M., Astrom, K.: Solutions to minimal generalized relative pose problems. In: OMNIVIS. (2005)
15. Geyer, C., Stewenius, H.: A nine-point algorithm for estimating para-catadioptric fundamental matrices. In: CVPR. (2007)
16. Li, H., Hartley, R.: A non-iterative method for correcting lens distortion from nine-point correspondenses. In: OMNIVIS. (2005)
17. Fischler, M., Bolles, R.: Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM **24** (1981) 381–395
18. Dhome, M., Richetin, M., Lapresté, J.T., Rives, G.: Determination of the attitude of 3-D objects from a single perspective view. PAMI **11** (1989) 1265–1278
19. Kukelova, Z., Bujnak, M., Pajdla, T.: Automatic generator of minimal problem solvers. In: ECCV. (2008)
20. Nistér, D.: A minimal solution to the generalized 3-point pose problem. In: CVPR. (2004)
21. Haralick, R., Lee, C., Ottenberg, K., Nolle, M.: Review and analysis of solutions of the three point perspective pose estimation problem. IJCV (1994)
22. Furukawa, Y., Ponce, J.: Accurate, dense, and robust multi-view stereopsis. PAMI (2009)
23. Furukawa, Y., Ponce, J.: Patch-based multi-view stereo software (200) http://grail.cs.washington.edu/software/pmvs.
24. Ramalingam, S., Lodha, S.: Adaptive enhancement of 3d scenes using hierarchical registration of texture-mapped 3d models. In: 3DIM. (2003)
25. Rusinkiewicz, S., Levoy, M.: Efficient variants of the icp algorithm. In: 3DIM. (2001)