3D Rigid Registration of Cad Point-Clouds

Conference Paper · March 2018

DOI: 10.1109/ICCSE1.2018.8373991

2 authors:

Ammar Hattab
Brown University
6 PUBLICATIONS **18** CITATIONS

SEE PROFILE

Gabriel Taubin
Brown University
231 PUBLICATIONS **11,417** CITATIONS

SEE PROFILE

3D Rigid Registration of CAD Point-Clouds

Ammar Hattab
School of Engineering
Brown University
Providence, RI, USA
ammar_hattab@brown.edu

Gabriel Taubin
School of Engineering
Brown University
Providence, RI, USA
gabriel_taubin@brown.edu

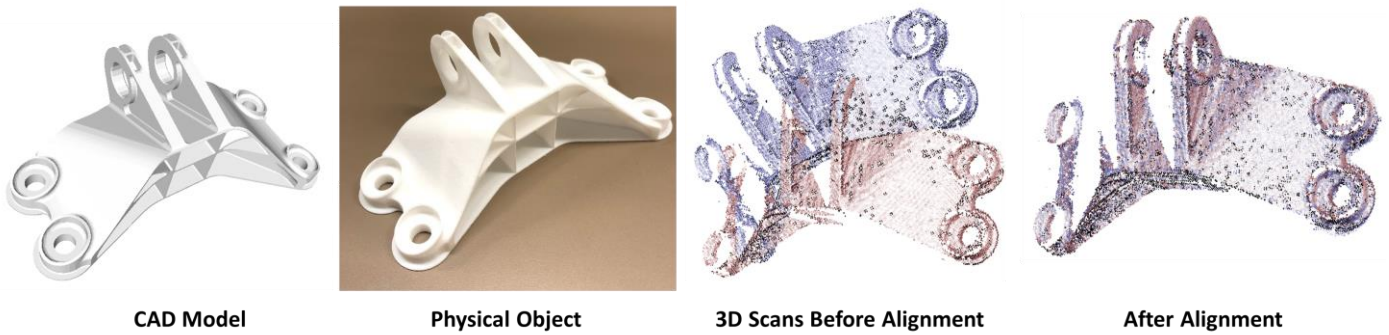

Figure 1: An example CAD object, 3D scanned from two different viewpoints, and aligned using our method

***Abstract*— In this paper, we introduce a new method for the rough alignment of point-clouds. We focus on a special type of point-clouds that is composed of simple geometric shapes like planes, cylinders, cones, etc. We call them 3D CAD point clouds. They are usually used in industrial and mechanical applications. The proposed method starts by detecting basic shapes in the point-clouds. And then using them to find the best transformation (rotation and translation) that aligns the point-clouds. Then, we run the fine alignment step using the iterative closest point method (ICP). We show several real-world examples of point-clouds before and after the alignment using this method. The results suggest that the proposed method works well in most cases given enough overlap between the point-clouds.**

Keywords—3D Coarse Registration; CAD Point Clouds; Shapes Detection;

I. INTRODUCTION

3D scanners are used to capture the shape of a real-world object to produce a 3D virtual model on the computer. The process usually involves capturing 3D data (in the form of a 3D point-cloud) of the same physical object from multiple views. Each view is selected to cover the object from a certain direction to capture the whole object. But, the views are usually not related to each other. So, it's important to align these views to produce one merged point cloud that could be then reconstructed into a 3D surface.

The alignment step is not a trivial task. And it usually needs enough overlap between the two views, such that we could use the overlapping parts of the point-clouds to find the best alignment. Most often, the alignment step is formulated as an optimization problem that depends on the geometry of the point-clouds.

If we have some prior knowledge about the geometry of the point clouds, can we exploit it to solve the alignment problem? In this work, we focus on a certain type of objects coming from industrial and mechanical applications. Those objects which are designed by CAD systems, and tend to be composed of simple geometric shapes like planes, cylinders, cones, etc.

In this paper, we assume that we have two point-clouds representing two different views of a CAD object. We propose an algorithm that first detects the basic shapes in the two point-clouds. Then it searches for the best combination of basic shapes that gives the best alignment. Finding the best alignment problem is divided into two sub-problems, first, finding the best rotation and then, finding the best translation. At last a few iterations of the iterative closest point "ICP" algorithm [12] are applied to get a fine registration between the two point-clouds, see Figure 1 for an example.

II. PREVIOUS WORK

Due to its importance, there's been a lot of research into the problem of 3D registration. Refer to [5] and [6] for a comprehensive survey on the topic.

In general, registration problems can be divided into rigid and non-rigid registration. In this work, we focus on the rigid registration problem, where the point-clouds represent different views of the same rigid object that has not changed or deformed.

Rigid alignment usually involves two stages: coarse alignment and fine alignment.

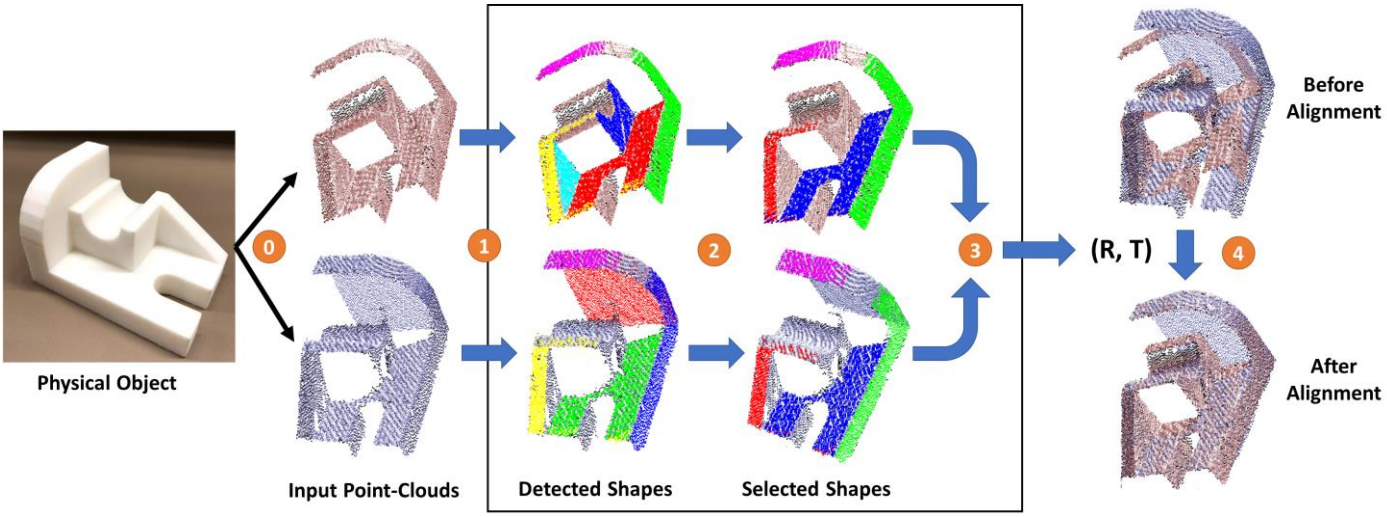

Figure 2: Method Steps: 0) 3D scanning the physical object to get two point-clouds representing two views. 1) Detecting basic shapes like planes and cylinders in the two point-clouds, each shape points are colored uniquely. 2) Find the best combination of the detected shapes. Corresponding shapes are given the same color. 3) Finding the best rotation R and the best translation T (the output of the method). 4) Using R, T to align the two point-clouds and then applying ICP algorithm for fine registration.

The goal of the coarse alignment step (often called global registration) is to roughly align the two point-clouds to prepare them for the fine registration step. A variant of the iterative closest point (ICP) [12-13], is used for the fine alignment, but it needs a good initial rough alignment, otherwise, it can get stuck in a local minima.

Previous methods for rough alignment include PCA [7], features-based [8] (like high curvature features or using sharp edges), global search algorithms [10], and using planes [11, 14-16].

PCA is the simplest method for rough alignment. It tries to align the principal directions of the two point-clouds, but it fails if the dimensions of the point-cloud are close to each other. Search algorithms usually take a long time to test different possible solutions. In the case of CAD point clouds, edges and high curvature points are usually lost and are hard to retrieve, so it's hard to use them for registration.

In this work, we believe that the easiest features to extract from CAD point clouds are the basic shapes surfaces like planes and cylinders because they cover a large surface area. Some previous papers used planes especially for camera tracking in SLAM applications [10]. The difference is that in SLAM applications there are usually plenty of planes detected from the walls and floors, but in our case, for CAD point clouds it's not enough to use planes only to align them. Especially with partial views, fewer planes are usually detected, and more importantly detected planes should be shared between two or more views. So, we need also to use other basic shapes like cylinders as they restrict the search domain and reduce the transformation degrees of freedom more than the planes. In our method, we define one simple formulation for the different types of basic shapes.

III. Method

The algorithm takes as an input two partial point clouds captured of the same object from two different views, and the output is the rigid transformation matrix that aligns one of them to the second, See Figure 2. We will first talk about a manual approach for the alignment, then we will propose the automated approach.

A. Manual Approach

Although there are several automated coarse alignment algorithms; people more often prefer to do it manually. Automated algorithms fail in many cases, or take too long to search for the optimal alignment. The conventional manual coarse alignment is done by manually selecting a few corresponding points between the two point-clouds. Then finding the best transformation between those points.

In the case of CAD point clouds, there are no clear distinct points because the high curvature points are usually lost in the point clouds. So, even this manual approach fails in many cases. In this work, we present a better manual approach as in the following steps:

1) Step 1: Manually Selecting Basic Shapes:

We allow the user to manually select a number of corresponding basic shapes in the same order in the two point-clouds. First, the user should specify the shape type (plane, cylinder, etc.), then he should click anywhere on the surface of that basic shape. Then our software will use a KD-tree to find the nearest K neighbors of the clicked point, then find the best fitting shape (of the type specified) that fits these points in the least squares sense. The fitting is done by minimizing the least squares distance from the points to the surface of the specified shape according to [1].

After the initial fitting of the surface, we run a few iterations to refine the fitting. In each iteration, we first find the inlier points to that shape of the whole point cloud using a specific distance and angle thresholds and remove the outlier points, and then we run the least squares fitting again only to the inlier points. After a few iterations, the refinement loop stops on the best fitting shape at that specific point.

The user should repeat this operation a few times, selecting shapes by clicking on the two point-cloud in the same order to select corresponding shapes. After that, the algorithm will use those selected shapes to find the best transformation that aligns the two point-clouds.

2) Step 2: Find Best Rotation:

The outputs of the previous step are two lists of basic shapes (planes, cylinders, etc.), one for each point cloud. We divide the problem of finding the best transformation into two sub-problems: finding the best rotation that aligns the two lists of shapes, then finding the best translation (see Figure 3 below).

We could use each corresponding pair of shapes to define a rotation that aligns the two shapes together, let's assume that we have N pairs of shapes. That means we have N rotations. Instead, we want to find one rotation that best aligns all the shapes of the first point-cloud to their corresponding shapes of the second point-cloud. Where each pair of shapes contributes to this rotation.

So, in general, each pair of shapes contribute with a pair of vectors (v_i, v_i') that should be aligned together. For planes we want to align the normal of the planes, so the pair of vectors could be the pair of planes normals. For cylinders, cones and toruses, we want to align their axis of rotation, so the pair of vectors to be aligned are the pairs of axis of rotation of the two corresponding shapes. Spheres don't contribute to the problem of finding best rotation, but only for finding the best translation.

We define our objective as to find the rotation matrix R that minimizes the squares of distances between N pairs of vectors:

$$E = \sum_{i=1}^{N} \|v_i' - Rv_i\|^2$$

Since each vector could be considered as a point in the unit sphere, we could use the same formulation that's used to align corresponding points as in ICP algorithm. So, we used the SVD approach of [2] to find the best rotation matrix R.

3) Step 3: Find Best Translation:

After finding the best Rotation Matrix R, we use it to rotate the second point cloud and all its detected shapes to be in the same orientation of the first point cloud. All that remains is to translate the second point-cloud to match the first point-cloud.

To find the best translation vector T, we define a separate minimization problem, that minimizes the sum of squared orthogonal distances between each pair of basic shapes of the selected shapes.

$$E = \sum_{i=1}^{n} (T.N_i - D_i)^2$$

Where N_i is the orthogonal direction vector of the two shapes (for example in the case of plane we can set it to the average of the two corresponding planes normals after rotation). While D_i is the orthogonal distance between the two shapes.

To minimize it with respect to T, we set:

$$\frac{dE}{dT} = \sum N_i(T.N_i - D_i) = 0$$

This defines the following system of equations: $AT = b$

$$\begin{bmatrix} \sum N_{ix}{}^2 & \sum N_{ix}N_{iy} & \sum N_{ix}N_{iz} \\ \sum N_{iy}N_{ix} & \sum N_{iy}{}^2 & \sum N_{iy}N_{iz} \\ \sum N_{iz}N_{ix} & \sum N_{iz}N_{iy} & \sum N_{iz}{}^2 \end{bmatrix} \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} = \begin{bmatrix} \sum N_{ix}D_i \\ \sum N_{iy}D_i \\ \sum N_{iz}D_i \end{bmatrix}$$

Then we fill those matrices (A, b) by looping through each pair of selected shapes. Then we find the best translation as:

$$T = A^{-1}b$$

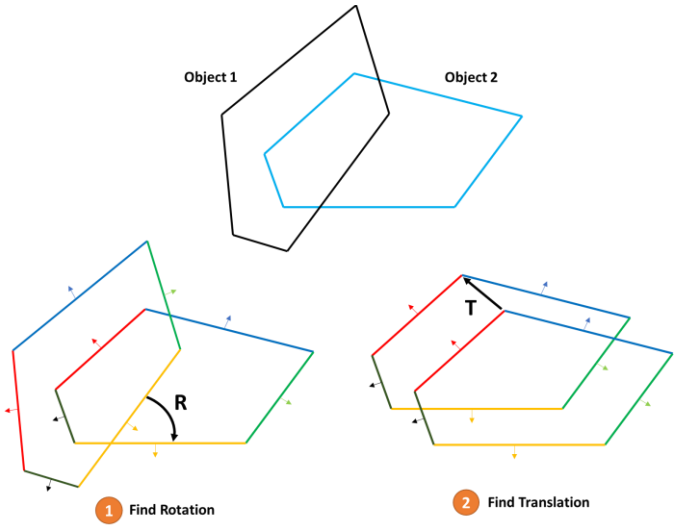

Figure 3: Finding the best alignment problem is divided into two sub-problems: 1) Find the best rotation 2) find the best translation.

4) Step 4: Fine registration using ICP:

After the coarse registration defined above, we refine the alignment using ICP to get a fine registration between the two point-clouds.

B. Automated Approach

Instead of asking the user to manually select the shapes of the two point-clouds, we could automate the operation of detecting those shapes as in the following steps:

1) Step 1: Detect basic shapes using RANSAC:

There are several methods to detect basic shapes in point clouds, we could simply select random points and apply the least squares fitting and refining approach described in step 1 above. Instead, we used the RANSAC method in [3] to detect

the basic shape. Their approach works by randomly selecting points in the point-cloud and testing whether they form one of the basic shapes. Then out of many iterations, they keep the detected shapes that has many inlier points. See an example in Figure 4 below.

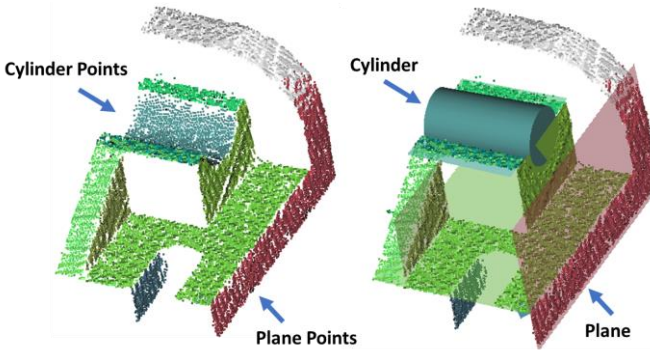

Figure 4: Detected shapes example. Left: inlier points for each shape are given different color. Right: some examples of the detected shapes.

2) Step 2: Find the correct combination and ordering between the detected shapes

The shapes are detected in the two point-clouds in arbitrary order. We need to get the correct correspondence between the detected shapes to run the algorithm successfully. Even though we have a few detected shapes, it's still not a trivial task to find the correct correspondence. Because the number of possible permutations grows exponentially. It might not be efficient to try every possible permutation, so we choose to use RANSAC algorithm to do that.

In each iteration of the algorithm, we randomly select three shapes of each point-cloud in a random order. The shapes selected should be of the same type (match planes to planes, cylinders to cylinders, etc.). and we test if the selected combination gives the best alignment. And after many iterations, we select the combination that gives the best alignment. Then we run the steps described above to find the best rotation and translation.

To test the randomly selected combination in each iteration we first calculate the best transformation between the selected shapes (as described above) and using this transformation we align the shapes. Then we calculate the rotation and translation error between the detected shapes. This test is very efficient, so we use it to excludes most wrong combinations. Still, some wrong combinations cannot be excluded using this test (See Figure 5). So, we must apply another test, which's to align the two point-clouds and compute the actual total alignment error between them (by calculating the distance from each point on one point-cloud to the nearest point in the second point cloud).

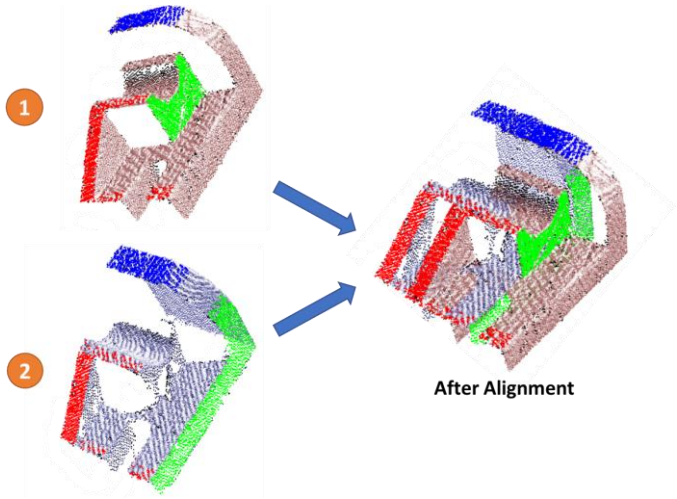

Figure 5: An example of a wrong random combination of basic shapes. After alignment the orthogonal distance between the basic shapes (the red, green and blue planes in this case) is almost zero, but the total distance (point to point) between the two point-clouds is large.

IV. Results and Discussion

We used a low-resolution Optoma projector and a Basler camera to perform gray code structure light scanning using the method of [4]. For several CAD objects, we captured a 3D scan from multiple views and then we used the method described above to align the resulting point clouds of each view. Figure 6 shows several alignment examples, while (table 1) summarizes the resulting average alignment errors of each example.

TABLE I. Registration Results

	Rotation Error	***Translation Error***	***Total Error (point-to-point)***	***Average Iterations***
1	0.0628	0.00591	1.9229	<8000
2	0.0290	0.1660	0.8610	<1000
3	0.0267	0.0000	0.6649	<1000
4	0.0514	1.7911	0.8208	<1000
5	0.0509	0.0112	0.4263	<1000

The numbers in the table are calculated by running the detection algorithm many times on each model and then taking the average of each. The last column calculates the average number of RANSAC iterations the algorithm took to find the correct combination of basic shapes.

What happens if there are not enough shapes detected? For example, sometimes only two detected planes are shared among the two views. Or even three planes with two of them parallel to each other. We noticed that ICP often works successfully even if the initial alignment it's too rough. For example, it will still succeed if we run it on the result of Figure 5.

Another option for future research is to formulate the problem as a restricted ICP algorithm. Such that we restrict the

possible transformations returned from ICP to keep the detected basic shapes aligned with each other.

At last, we noticed that the method works well even if the object has been physically changed between the two point-clouds as in example 4 in Figure 6.

V. Conclusion

In this paper, we presented a method to align CAD point clouds by detecting basic shapes (planes, cylinders, cones, etc) in them and matching the shapes to find the best possible alignment. We noticed that the algorithm works well when there are enough overlapping basic shapes between the two views. If there are no enough shapes detected, in many cases the algorithm provides a good rough alignment to start the ICP algorithm.

Acknowledgment

The work described herein was partially supported by a Brown Fellowship and by NSF grant IIP-1500249.

References

[1] Ahn, Sung Joon. Least squares orthogonal distance fitting of curves and surfaces in space. Vol. 3151. Springer Science & Business Media, 2004.

[2] Arun, K. Somani, Thomas S. Huang, and Steven D. Blostein. "Least-squares fitting of two 3-D point sets." IEEE Transactions on pattern analysis and machine intelligence 5 (1987): 698-700.

[3] Schnabel, Ruwen, Roland Wahl, and Reinhard Klein. "Efficient RANSAC for point-cloud shape detection." Computer graphics forum. Vol. 26. No. 2. Blackwell Publishing Ltd, 2007.

[4] Moreno, Daniel, and Gabriel Taubin. "Simple, accurate, and robust projector]-camera calibration." 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on. IEEE, 2012.

[5] Pomerleau, François, Francis Colas, and Roland Siegwart. "A review of point cloud registration algorithms for mobile robotics." Foundations and Trends® in Robotics 4.1 (2015): 1-104.

[6] Tam, Gary KL, et al. "Registration of 3D point clouds and meshes: a survey from rigid to nonrigid." IEEE transactions on visualization and computer graphics 19.7 (2013): 1199-1217.

[7] Liu, Yu-Shen, and Karthik Ramani. "Robust principal axes determination for point-based shapes using least median of squares." Computer-Aided Design 41.4 (2009): 293-305.

[8] Gal, Ran, and Daniel Cohen-Or. "Salient geometric features for partial shape matching and similarity." ACM Transactions on Graphics (TOG) 25.1 (2006): 130-150.

[9] Chen, Chu-Song, Yi-Ping Hung, and Jen-Bo Cheng. "RANSAC-based DARCES: A new approach to fast automatic registration of partially overlapping range images." IEEE Transactions on Pattern Analysis and Machine Intelligence21.11 (1999): 1229-1234.

[10] Yang, Jiaolong, et al. "Go-ICP: a globally optimal solution to 3D ICP point-set registration." IEEE transactions on pattern analysis and machine intelligence 38.11 (2016): 2241-2254.

[11] Xiao, Junhao, Bejamin Adler, and Houxiang Zhang. "3D point cloud registration based on planar surfaces." Multisensor Fusion and Integration for Intelligent Systems (MFI), 2012 IEEE Conference on. IEEE, 2012.

[12] Besl, Paul J., and Neil D. McKay. "A method for registration of 3-D shapes." IEEE Transactions on pattern analysis and machine intelligence 14.2 (1992): 239-256.

[13] Chen, Yang, and Gérard Medioni. "Object modelling by registration of multiple range images." Image and vision computing 10.3 (1992): 145-155.

[14] Cupec, Robert, et al. "Global localization based on 3D planar surface segments." arXiv preprint arXiv:1310.0314 (2013).

[15] Theiler, P. W., and K. Schindler. "Automatic registration of terrestrial laser scanner point clouds using natural planar surfaces." ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences 3 (2012): 173-178.

[16] Pathak, Kaustubh, et al. "Online three-dimensional SLAM by registration of large planar surface segments and closed-form pose-graph relaxation." Journal of Field Robotics 27.1 (2010): 52-84.

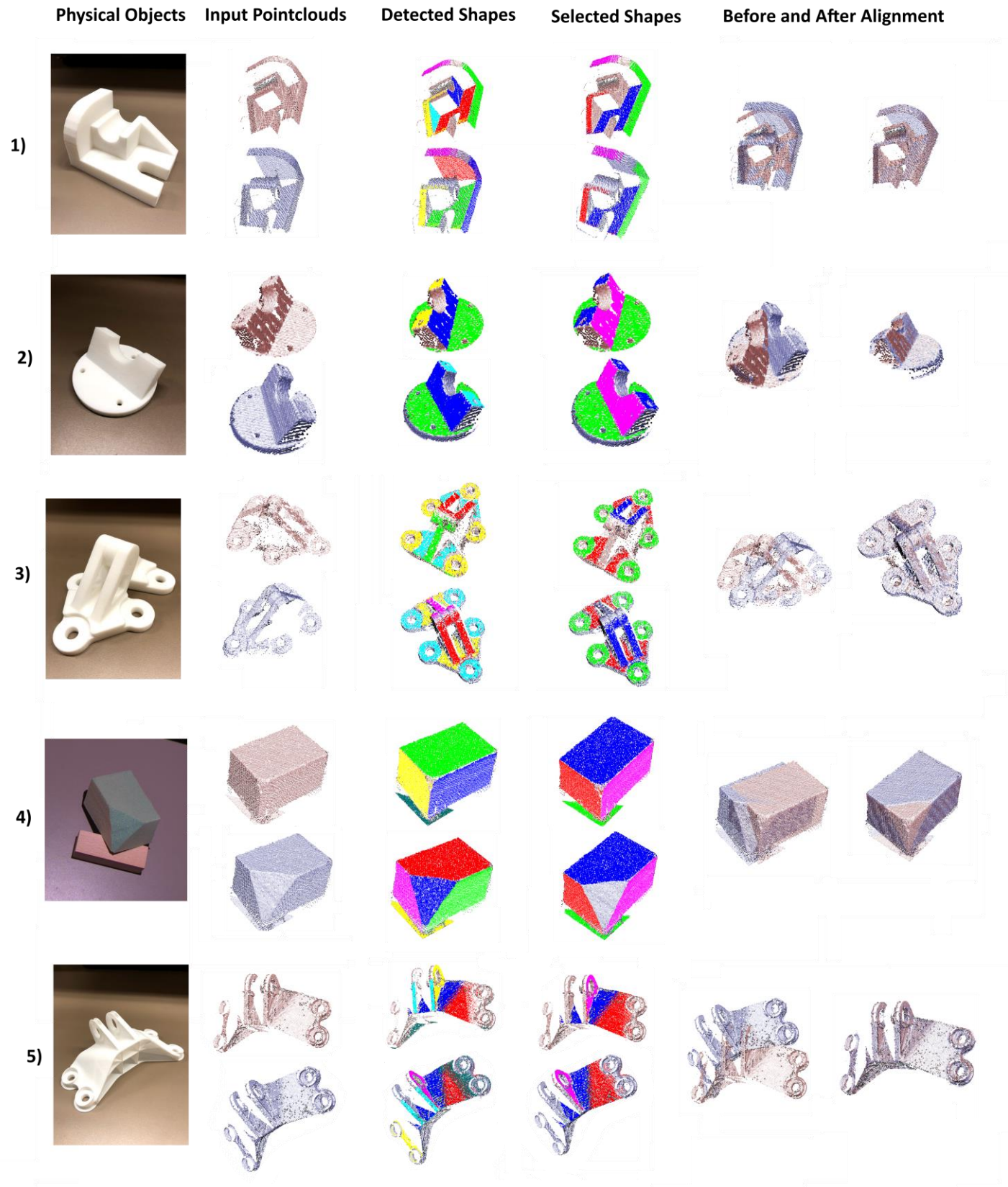

Figure 6: Five examples of aligning CAD point-clouds.