Final Report
Semester Project
Learning Algorithms and Systems Laboratory

Inferring Assembly Objects and Sequences from Demonstrations

Supervisor:
Prof Aude Billard

Assistant:
Dr Athanasios Polydoros

Author:
Mahdi Nobar

11th June, 2019

Contents

1 Introduction

1.1 Motivation

Autonomous manufacturing is an interesting research field that can benefit from Machine Learning approaches. *Traditional* control techniques can be combined with artificial intelligence in order to allow robots to learn new behaviours through demonstration. Researchers at Siemens Corporate Technology in United States have developed a set of gears to test different robot learning approaches for the assembly process as shown on figure (1). The assembly of these gears requires high precision and the ability to learn changing complex dynamics. [8]

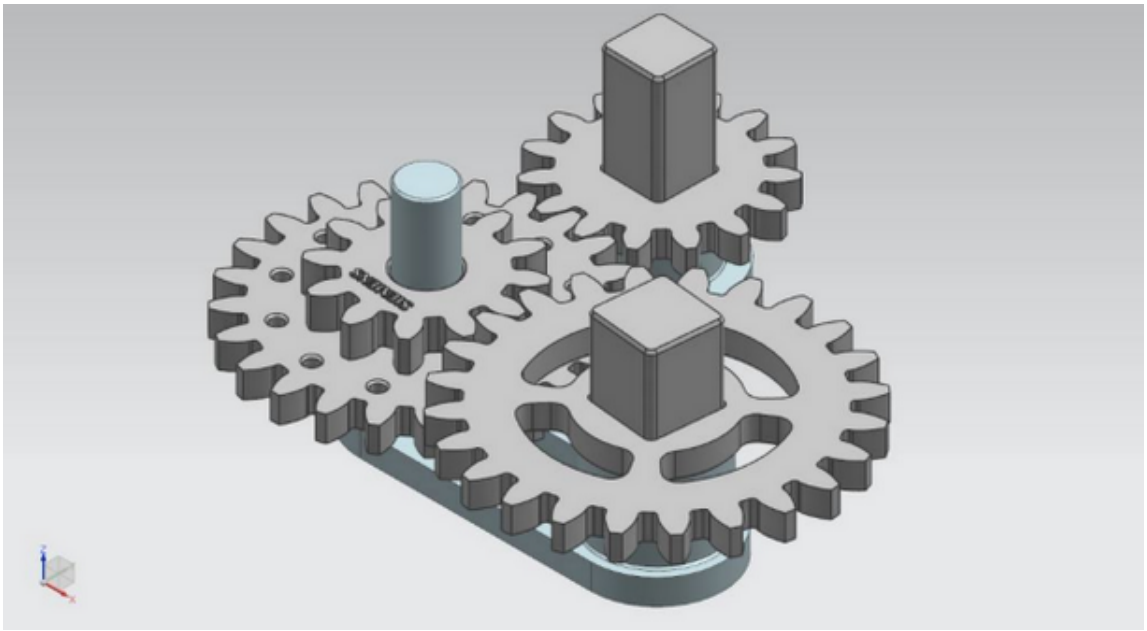


Figure 1: Set of gears proposed by Siemens Corporate Technology in Berkeley, California as benchmark for learning approach developments

In fact, traditional control and Artificial Intelligence approaches are combined to increase the automation flexibility in tasks such as assembly; because robot learning covers the methodology, theory and art of enabling a robot, or any other automation system, to learn new skills and adapt to a flexible environment. Also computer vision techniques are used in industrial robots. In fact, vision guided robots can be the potential application of the algorithm developed in this project.

1.1.1 Vision Guided Robots

A VGR[1] System is basically a robot fitted with one or more cameras used as sensors to provide a secondary feedback signal to the robot controller to more accurately move to a variable target position [16]. VGR is rapidly transforming production processes by enabling robots to be highly adaptable to the new production line set up, while dramatically reducing

[1] Vision Guided Robot

the cost and complexity of fixed tooling previously associated with the design and set up of robots. So the overall benefits of VGR are listed below:

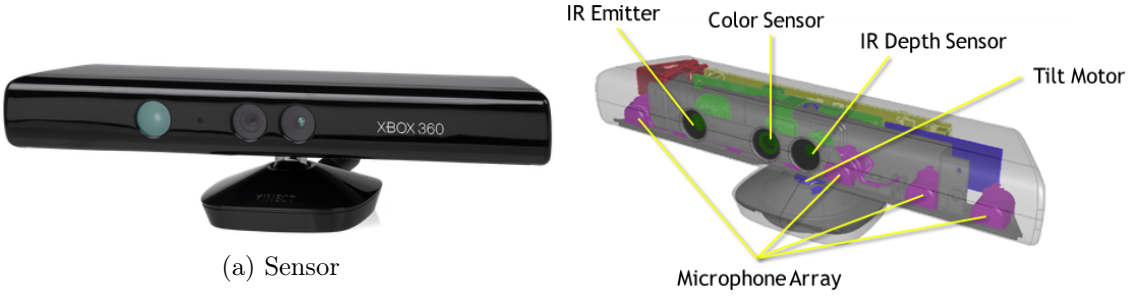
- Switching between products and batch runs is software controlled and very fast, with no mechanical adjustments
- High residual value, even if production is changed
- High machinery efficiency, reliability, and flexibility
- Possibility to integrate a majority of secondary operations such as deburring, clean blowing, washing and measuring
- Reduces manual work

1.2 Tools

In this project, a sensor is needed to provide color and depth data. Thus, Kinect Xbox is selected (section 1.2.1). Moreover, for the implementation of the machine learning algorithms, Scikit Learn framework is used (section 1.2.2). Besides, OpenCV and Open3D libraries are utilized because they allow to process the acquired image and three dimensional data (sections 1.2.3 and 1.2.4).

1.2.1 Kinect Xbox

The Kinect sensor is a horizontal bar connected to a small base with a motorized pivot and is designed to be positioned lengthwise above or below the video display as shown on figure(2a). The device features a 3D Depth sensor (IR Emitter with IR Camera and Depth Sensor), RGB camera (Color Sensor), Microphone array and Tilt motor shown on figure (2b).[10] Here, it is utilized to acquire RGB image with its corresponding depth sensor data.



(a) Sensor

(b) Components

Figure 2: Microsoft Kinect Xbox Version 1

1.2.2 Scikit Learn

There are several Python libraries which provide solid implementations of a range of machine learning algorithms. One of the best known is Scikit-Learn, a package that provides efficient versions of a large number of common algorithms. Scikit-Learn is characterized

by a clean, uniform, and streamlined API, as well as by very useful and complete online documentation.[9] In this project, Scikit-learn is used for clustering and data pre-processing.

1.2.3 OpenCV

OpenCV (Open source computer vision) is a library of programming functions mainly aimed at real-time computer vision.[12] It is used to detect contours and modify images.

1.2.4 Open3D

Open3D is an open-source library that supports rapid development of software that deals with 3D data. The Open3D front-end exposes a set of carefully selected data structures and algorithms in both C++ and Python. The back-end is highly optimized and is set up for parallelization. [21] Here Opne3D is used for image registration, point cloud extraction using RGB image with depth sensor data and camera calibration.

2 Theory

In this section an overall explanation of key aspects of the theoretical concepts that are used in this project is provided. In fact, it is focused on the concepts which are used in order to build an algorithm to infer assembly of two Siemens challenge objects introduced on previous section. First, data are standardized in order to utilize them for unsupervised learning. Next, based on intrinsic calibration parameters of Kinect Xbox sensor for its RGB images, the corresponding point cloud of each RGB frame with its depth sensor data is required to be computed. Then initially a contour around each component is drawn in order to detect number of objects for clustering. Furthermore, in order to reduce the noise effect at acquired data, a technique based on image registration is used, and then Fast Point Feature Histograms are utilized to replace the (x,y,z) and color features. In fact, Fast Point Feature Histograms provide features which are invariant to the position and orientation of the objects, so it is used to reduce the noise effect on section 3.5. Finally, a metric to measure the difference between distributions is introduced. This measure is used to infer the assembled objects by comparison of similarity between assigned clusters to each objects.

2.1 Data Standardization

In order to train a model based on features with different order of magnitude or different dimensions it is required to standardize data. For example in this project, the dimension of (x, y, z) features is different from the dimension of the RGB color features,and the order of magnitude of the depth sensor data is different from the order of magnitude of (x, y) features. Data standardization is applied by removal of the mean value of each feature and scaling the remaining by corresponding standard deviation.[15]

2.2 Contours

In object detection, contour is a curve which joins all of the continuous points along the boundary in an image. The boundary is defined by the sharp difference between the pixel values. In fact, contours are the boundaries of a shape with same intensity. For example, in this project the input frame is transferred to a gray-scale image then a threshold is applied to distinguish the components from each other and the background. Then the threshold result of that gray-scale image is used to distinguish the boundary of the components and draw the corresponding contours. [1] Therefore, in order to increase the detection accuracy of the boundaries, the binary equivalence of the image is used.[19]

2.3 Gaussian Mixture

A Gaussian mixture model is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. One can think of mixture models as generalizing k-means clustering to incorporate information about the covariance structure of the data as well as the centers of the latent Gaussians.[4] Initially, GMM is mainly utilized to cluster the acquired data. The aim was to infer the objects by using evaluation criteria of the unsupervised learning to fit Gaussians on each objects. So it was expected that the optimum number of required Gaussians would be equal to the number of objects. However, because of the facts that data are noisy and components are too small and close to each other this approach for utilization of GMM was unsussessful as is described in section 4.1. Then based on the results, only one Gaussian model is fit to each detected objects by computer vision techniques. Afterwards, the Gaussian model is used to infer the assembly of any object pairs.

2.4 Point Feature Histograms

The goal of Point Feature Histograms (PFH) formulation is to encode a point's k-neighborhood geometrical properties by generalizing the mean curvature around the point using a multi-dimensional histogram of values.[14] This highly dimensional hyperspace provides an informative signature for the feature representation, is invariant to the 6D pose of the underlying surface, and copes very well with different sampling densities or noise levels present in the neighborhood.

A Point Feature Histogram representation is based on the relationships between the points in the k-neighborhood and their estimated surface normals. Simply put, it attempts to capture as best as possible the sampled surface variations by taking into account all the interactions between the directions of the estimated normals. The resultant hyperspace is thus dependent on the quality of the surface normal estimations at each point.

Figure (3) presents an influence region diagram of the PFH computation for a query point (p_q), marked with red and placed in the middle of a circle (sphere in 3D) with radius r, and all its k neighbors (points with distances smaller than the radius r) are fully interconnected in a mesh. The final PFH descriptor is computed as a histogram of relationships between all pairs of points in the neighborhood, and thus has a computational complexity of $O(k^2)$.

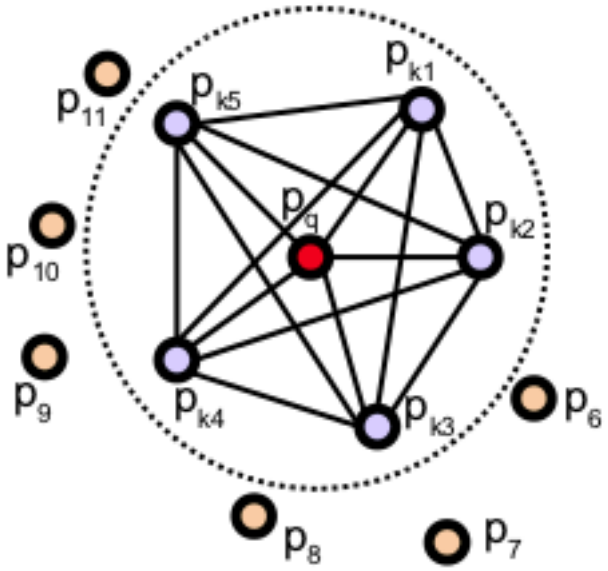


Figure 3: *Influence region diagram of point feature histogram*

To compute the relative difference between two points p_i and p_j and their associated normals n_i and n_j, we define a fixed coordinate frame at one of the points (look at figure (4)).

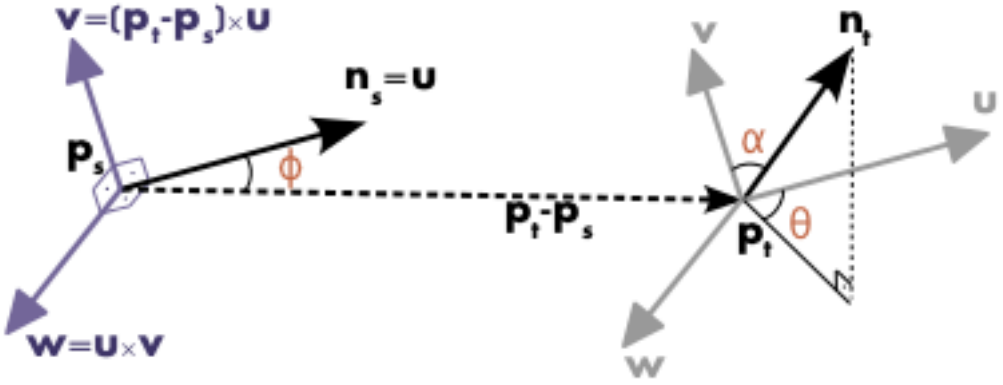


Figure 4: ***uvw*** *coordinate system between two points inside influence region*

Using the above **uvw** frame, the difference between the two normals n_s and n_t can be expressed as a set of angular features as follows:

where d is the Euclidean distance between the two points $\boldsymbol{p}_s$ and $\boldsymbol{p}_t$, $d = \|\boldsymbol{p}_t - \boldsymbol{p}_s\|_2$. The quadruplet $\langle \alpha, \phi, \theta, d \rangle$ is computed for each pair of two points in k-neighborhood, therefore reducing the 12 values (xyz and normal information) of the two points and their normals to 4.

To create the final PFH representation for the query point, the set of all quadruplets is binned into a histogram. The binning process divides each features's value range into b subdivisions, and counts the number of occurrences in each sub-interval. Since three out of the four features presented above are measure of the angles between normals, their

values can easily be normalized to the same interval on the trigonometric circle. A binning example is to divide each feature interval into the same number of equal parts, and therefore create a histogram with b^4 bins in a fully correlated space. In this space, a histogram bin increment corresponds to a point having certain values for all its 4 features. So the values of the bin with maximum height is chosen as PHF representation of the query point.

2.5 Fast Point Feature Histograms

The theoretical computational complexity of the Point Feature Histogram as described in previous section for a given point cloud P with n points is $O(nk^2)$, where k is the number of neighbors for each point p in point cloud P. For real-time or near real-time applications, the computation of Point Feature Histograms in dense point neighborhoods can represent one of the major bottlenecks.

Fast Point Feature Histograms (FPFH) is the simplified version of point feature histogram which reduces the computational complexity of the algorithm to $O(nk)$, while still retaining most of the discriminative power of the PFH.[2]

First, for each query point p_q a set of tuples α, ϕ, θ between **only** itself and its neighbors are computed as based on the method described in last section. Then the calculated histogram is called Simplified Point Feature Histogram (SPFH). Thereafter according to equation (1), k neighbors of each point are re-determined, and the neighboring SPFH values are used to weight the final histogram of P_q (called FPFH).

$$FPFH(\mathbf{p}_q) = SPFH(\mathbf{p}_q) + \frac{1}{k}\sum_{i=1}^{k}\frac{SPFH(\mathbf{p}_k)}{\omega_k} \tag{1}$$

where the weight ω_k represents a distance between the query point p_q and a neighbor point p_k in some given metric space, thus scoring the (p_q, p_k) pair, but could just as well be selected as a different measure if necessary. To understand the importance of this weighting scheme, figure (7) presents the influence region diagram for a k-neighborhood set centered at p_q.

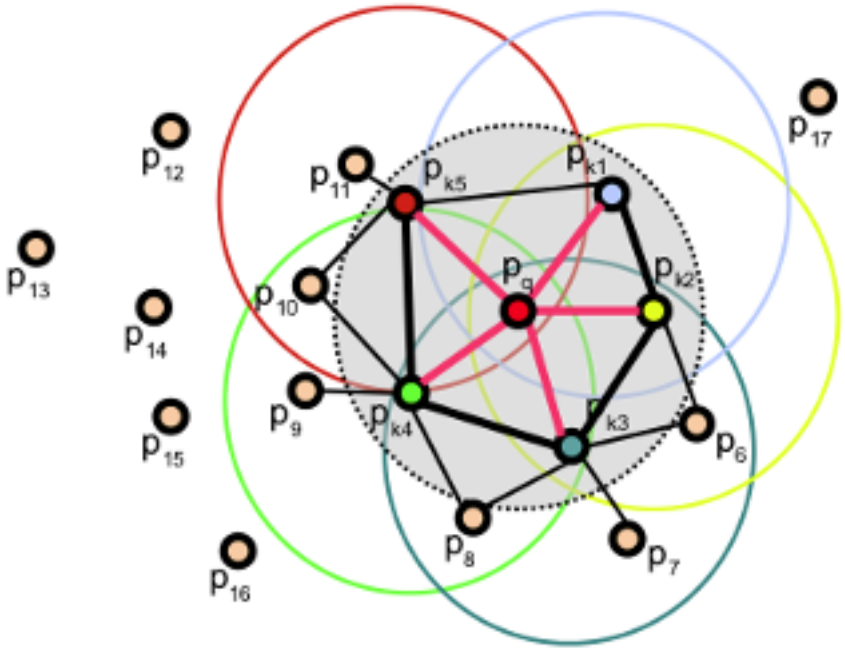


Figure 5: *Schematic representation of points involved in fast point feature histogram calculation for a query point* p_q

Therefore, for a given query point p_q, the algorithm first estimates its SPFH values by creating pairs between itself and its neighbors (illustrated using red lines). This is repeated for all the points in the dataset, followed by a re-weighting of the SPFH values of pq using the SPFH values of its p_k neighbors, thus creating the FPFH for p_q.

2.6 Image Registration

Image registration is the process of transforming different sets of data into one coordinate system.[7] Registration is necessary in order to be able to compare or integrate the data obtained from different measurements. In this project image registration is utilized in order to cancel out the noise effect which presents in captured frames. So herewith the theory behind two image registration techniques including global registration and iterative closest point registration is discussed. The utilization of these registrations is explained at *implementation* section.

2.6.1 Global Registration

Global registration refers to a class of registration algorithms which do not require an alignment for initialization. Therefore, these methods produce less tight alignment than local registration methods. Furthermore, the result of global registration can be utilized as initialization of the local methods.[5]

2.6.2 Iterative Closest Point Registration

Iterative Closest Point Registration or ICP registration is a local registration method. Local registration algorithms, in contrast with the global registration algorithms, require initial transformation to align roughly the source point cloud to the target point cloud.[6] Thus, ICP registration requires three inputs: source point cloud, target point cloud and the and an initial alignment transformation between them. Basically, ICP algorithm iterates over first finding the accordance set $K = (\mathbf{t}, \mathbf{s})$ from target point cloud $\mathbf{T}$, and source point cloud $\mathbf{S}$ which has been transformed with the current transformation matrix $\mathbf{T}$ at each iteration, and second updating the transformation $\mathbf{T}$ by optimization of an objective function $E(\mathbf{T})$ defined over the accordance set K. There are various options to choose the objective function. One of them which is used in this project is :

$$E(\mathbf{T}) = \sum_{(\mathbf{t},\mathbf{s})} \parallel \mathbf{t} - \mathbf{Ts} \parallel^2 \tag{2}$$

And the ICP registration which is based on cost function written on equation (2) is called *point-to-point* ICP algorithm, because the cost function minimizes the sum of Euclidean distance between the matched point from target point cloud with the transformation of its corresponding point at the source point cloud.

2.7 Kullback–Leibler Divergence

Kullback-Leibler divergence, also called relative entropy, is a measure to specify difference between two probability distribution. In other words, Kullback-Leibler divergence is a measure of surprise, and if it is equal to zero it means that the two distributions are identical. In contrast, large Kullback-Leibler divergence value indicates two distributions are more different from one another.[11] For discrete probability distributions P and Q defined on the same probability space, the Kullback–Leibler divergence is the expectation of the logarithmic difference between P and Q and is defined as equation (3):

$$D_{KL}(P \parallel Q) = -\sum_{(x \in \chi)} P(x)(\log(P(x)) - \log(Q(x))) \tag{3}$$

So if P and Q are both multi-variable normal distributions with means μ_0 and μ_1 and covariance matrices Σ_0 and Σ_1 with same dimension k, then relative entropy between them is defined as equation (4):

$$D_{KL}(P \parallel Q) = \frac{1}{2}(tr(\Sigma_1^{-1}\Sigma_0) + (\mu_1 - \mu_0)^T\Sigma_1^{-1}(\mu_1 - \mu_0) - k + \ln|\Sigma_1| - \ln|\Sigma_0|) \tag{4}$$

Equation (4) provides relative entropy measured in nat [1]. So if it is divided by $\ln 2$ it provides the divergence in bits.

3 Implementation and Explanation

In this project, the model proposed by Siemens is considered as a benchmark for the development of an inferring algorithm which combines computer vision techniques with

[1] natural unit of information

unsupervised learning to infer assembly process. Therefore, the components shown on figure (1) using a 3D printer with scale of 0.9 with respect to the original components were printed. So the results of the project can be used for development of inference capability of vision guided robots.

In this section, the models that have been developed to infer the assembly of two distinct objects are described. It is assumed that there are two input frames. First frame includes some number of assembly objects placed separate from each other, and second frame also includes the same objects but one object is either removed away or assembled with another object of first frame. According to the ultimate goal, the inference system should be able to detect the missing object and to infer which object has been merged with which one in second frame. So the inputs of the models are two RGB images with their corresponding depth sensor data, and the output is the assembled pair of objects.

3.1 Initial Models

Initially it was attempted to use data standardization, principle component analysis, clustering algorithms including Birch[1], GMM, DBSCAN[2], Mean-shift and K-means, and clustering performance evaluation like Calinski-Harabasz Index. The idea was to provide normalized and rich data for clustering algorithm, tune the hyper parameters and use evaluation index to choose the optimum number of clusters. It would be expected that the proposed idea would allow the system to infer the number of objects. However, based on he results that will be provided on next section this overall idea was replaced by proposal of models described next.

3.2 model_1

This model is the first model which could successfully infer the missing object and also detect if it has been merged with another object. This model can detect two assembled object except when the noise between data influences the results. Practically, the only drawback of this model is that it cannot detect if the small bar for the gears shown in figure (12a) has been merged with L shaped asymmetric object shown on figure (12c). Now step by step each main operations starting from the input frames is explained.

1. In order to apply morphological operation (i.e here erosion) the RGB image is converted to gray-scale image:
2. The gray-scale image is eroded by an appropriate kernel size in order to not allow small hallow inside gears to be detected as a contour:
3. Next a fixed-level threshold is applied to distinguish the boundary between the white background and objects, and the output is used to fit a closed contour to each objects.
4. Then the center of each contours are found and its corresponding (x, y, z) values on the point cloud are recorded.
5. Using a loop color and depth vales corresponding to all pixels outside each contours

[1]Balanced Iterative Reducing and Clustering using Hierarchies
[2]Density-Based Spatial Clustering of Applications with Noise

are put to a constant value[1]; therefore, at each time a frame with level depth and identical color outside of each contour is obtained.

6. After intrinsic calibration and having the point cloud of data, features are standardized to have zero mean and unit variance.
7. After points of background are removed, using full covariance matrix one Gaussian model is assigned to each object with initial mean point.

3.3 model_2

In this model a GMM[2] with number of clusters equal to the number of detected closed contours is trained. Also shade of object caused by illumination is removed by putting red color for all pixels inside each contour. Final model in fact is able to detect the objects but it is not capable of inferring which object has been assembled with which one; because the objects are too close to each other and there is noise and missing points at input data. This noise causes difference between one frame to another even if the position of object has remained unchanged. Therefore, noise deteriorates the clustering results.

3.4 model_3

In order to improve model_1[3] which has provided some desirable results, model_3 is also proposed. This model has been created based on model_1, but there are some key differences between them which are described below:

1. Surface normals are calculated.
2. FPFH[4] of point cloud are calculated which includes 33 features. In fact, any change in the position or orientation of the objects could influence the clustering results. So in particular, FPFH is used to make the features be invariant to the position and orientation of the objects.
3. Finally each Gaussian models are trained based on FPFH features.

This model can also infer the objects between two input frames. However, it cannot infer the pair of merged objects. Furthermore, there is also noise effect because if two frames taken from an identical scene are provided, then there is also unacceptable Kullback-Leibler divergence between the detected clusters which indicates that the problem of noise has not solve.

3.5 model_4

This model is also written based on model_1 which aims to improve its inference. There are some key modifications applied to model_4 which are:

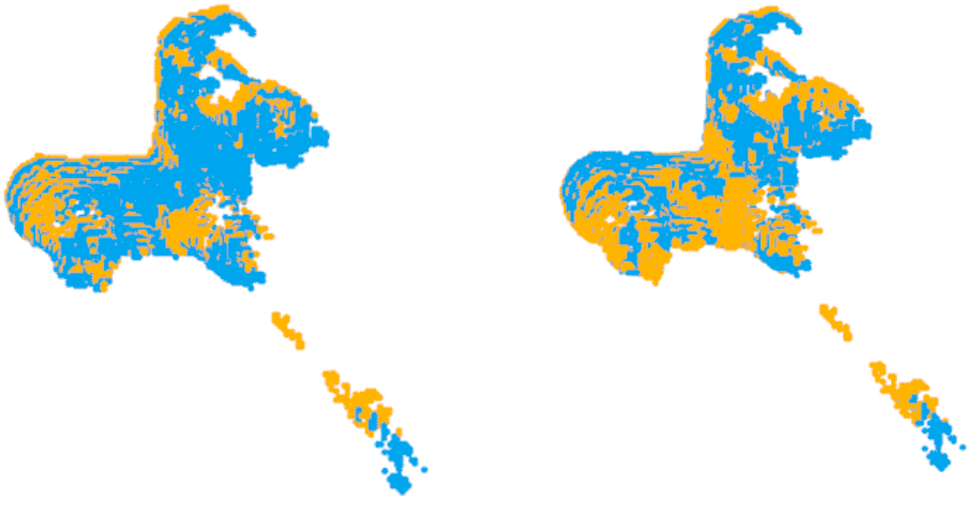
[1]Note that the constant value for depth should not be chosen equal to zero as point cloud generator function of Open3d framework removes the points with zero depth from point cloud so assignment of the initial mean point for each clusters would be technically problematic. For more information refer to the code.

[2]Gaussian Mixture Model

[3]Texts with the blue color corresponds to the code

[4]Fast Point Feature Histograms

1. Erosion is not used to remove out small contours detected for the hollow inner parts of gears. Instead, based on number of points of each contours, the ones with less points than a threshold are removed. In this way, only contours around the components remains and the contours are as tight as possible. This allows to not have background on fitting a Gaussian on each object, so the influence of level-depth background is minimized and it helps to improve cluster fitting to the data.
2. This model takes a couple of frames for the first and second scenes. So there are four frames give to this model as input. Each couple of frames taken from the scene but two times so they are different from each other because there exists noise at acquired data.
3. Global registration between each couple of frames are found. In order to find global registration a method called execute_global_registration is provided which also embeds down sampling of point cloud with an appropriate voxel size.
4. Then transformation matrix of point-to-point ICP registration is calculated. Afterwards, one of the frames among each couple is transformed by this result to decrease the noise effect by making the transformed point cloud be as close as possible to the second frame at the same pair of frames. In other words, a pair of point clouds $pc1$ and $pc2$ are considered, and local transformation T between them is calculated, and $pcd1$ is transformed by T to acheive the third point cloud $pcd3$ which is more similar to $pcd2$. As an example, the point clouds of initial couple of frames for one object before and after transforming point cloud of one of them using ICP registration is shown on figure (6).
5. Finally for data standardization only (x, y, z) features are standardized with respect to each other in order to not allow data related to color influence the input data given to the clustering algorithm.



(a) Point cloud of $pcd1$ in yellow with point cloud of $pcd2$ in cyan

(b) Point cloud of $pcd3$ in yellow with point cloud of $pcd2$ in cyan

Figure 6: Point cloud of pair of initial point clouds before and after transforming one of them by point-to-point ICP registration

According to (6), it is demonstrated that after transforming the first point cloud by point-to-point ICP registration the difference between the first and second point cloud decreases.

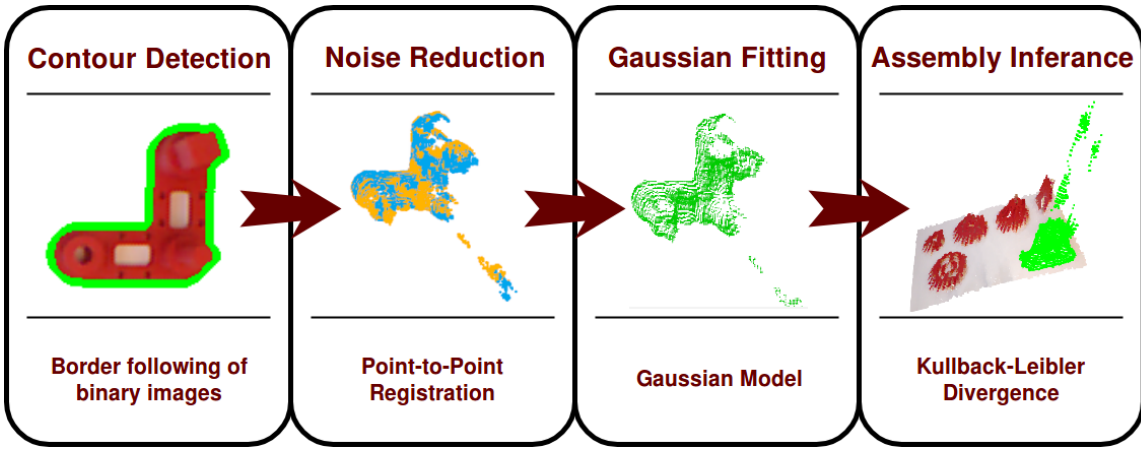


Figure 7: Schematic representation of key concepts used in model_4 represented for assembled parts shown on figure (17b)

3.6 Summary of the Implemented Models

To summarize the capabilities and characteristics of each models is provided on table (1). So any two objects that have been assembled could be detected by model_4 as soon as noise effect would decrease.

Name	Detection of all objects	Inferring any pair of assembled objects	Noise reduction
model_1	✓	✗	✗
model_2	✓	✗	✗
model_3	✓	✗	✗
model_4	✓	✓	✓

Table 1: Summary of capability of each proposed models

Also summary of key features used in each models is provided on table (2).

Name	Used methods
Initial models	• Data standardization • PCA[1] • Background removal • GMM, Birch, K-means, Mean-shift or DBSCAN
model_1	• Data standardization • Contours • Erosion • Removal of points outside contours • Mean point initialization • Gaussian distribution
model_2	• Data standardization • Contours • Erosion • Background removal • Removal of shade of objects • Mean point initialization • Gaussian distribution
model_3	• Data standardization • Contours • Erosion • Removal of points outside contours • FPFH features calculation • Gaussian distribution based on FPFH features
model_4	• Data standardization • Tight contours • Removal of points outside contours • Point-to-point ICP registration • Gaussian distribution

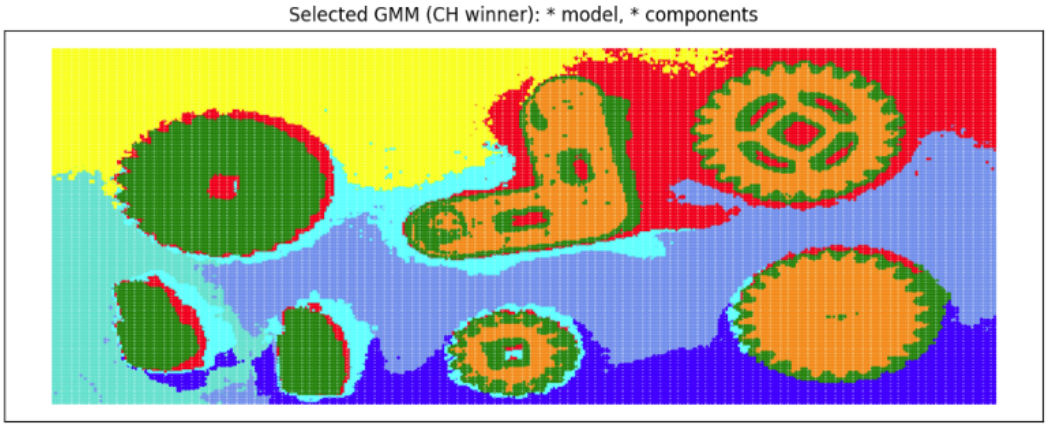
Table 2: Summary of key methods used in each proposed models

4 Results and Discussion

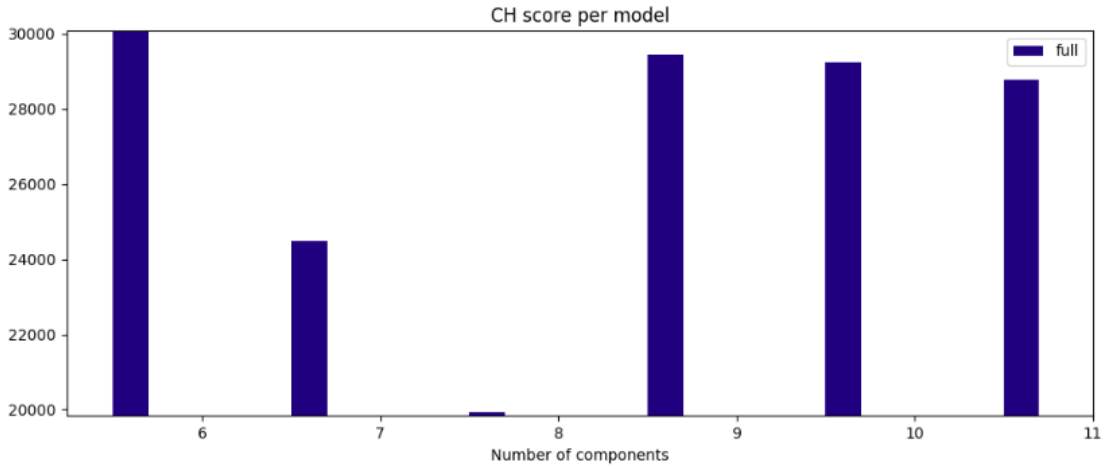
In this section the results of inference of the assembly of two objects is presented. First the results of initial attempts are qualitatively mentioned and described. Then the results of two scenarios with their explanation are provided. Finally, the noise reduction capability of model_4 is demonstrated.

4.1 Initial Attempts

Initially, it is attempted to detect number of components by evaluation criteria of unsupervised clustering. In other words, initially, GMM[1] was used to cluster data and based on clustering performance evaluation the optimum number of clusters to be calculated. It was expected that the optimum number of clusters would be equal to number of components plus one for the background as a separate cluster. However, this approach would provide incorrect results. As an example shown on figure (8), using GMM at the best performance provides totally wrong clustering. Part of this unsuccessful result was because of the fact that the background would influence the clustering.



(a) Clusters



(b) Calinski-Harabasz Index versus number of clusters for GMM

Figure 8: GMM results for the optimum K=8 based on Calinski-Harabasz index with principal component analysis with three features

Therefore in order to remove the background before using clustering to detect objects, GMM with pre defined two number of Gaussians was applied to data as shown on figure (9).

[1] Gaussian Mixture Model

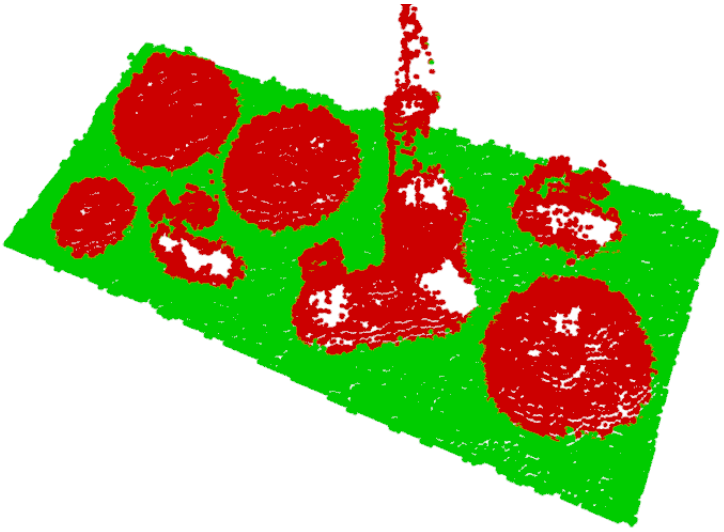


Figure 9: Point cloud of scene after clustering based on GMM with K=2

According to figure (9), GMM could distinguish the background from the components. So it was used to remove out the background. Next, the scene after background removal was provided to GMM to distinguish components and at the best result it could roughly distinguish objects as shown on figure (10). This figure demonstrates the fact that GMM could not detect correctly number of components. This would be because of the facts that either components are too small and close to each other that it would influence the detection or there is noise at data and/or randomly missing depth sensor data. Also the shadow around gear teeth would influence the results.

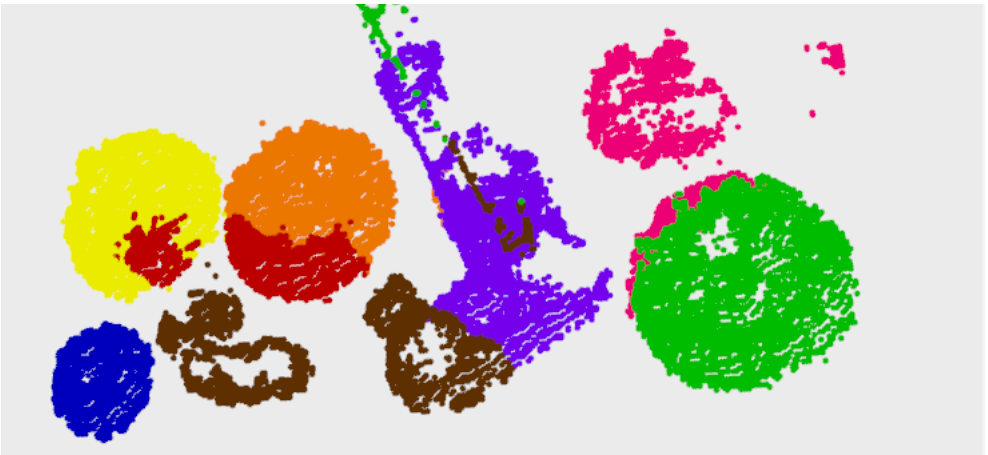
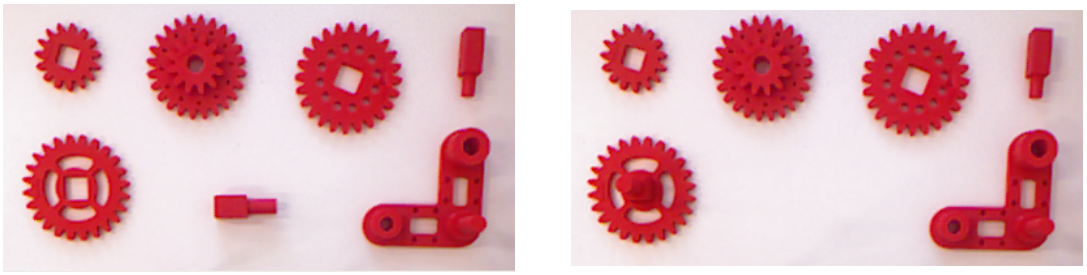


Figure 10: Point cloud of scene after background removal and clustering points based on GMM using only (x,y) features with optimum K=8 based on Calinski-Harabasz Index

4.2 Scenario One

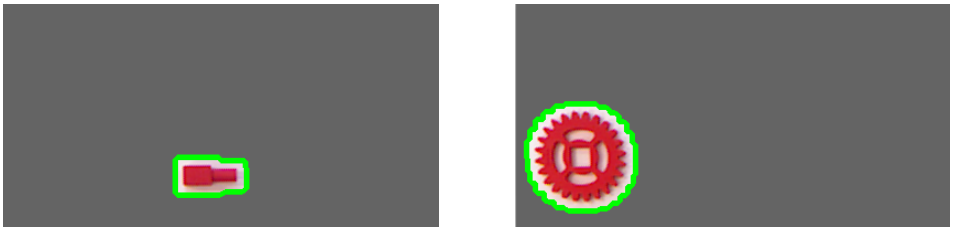
In this scenario first frame includes all seven components separately put on a table as shown on figure (11a). Second frame shown on figure (11b) includes the same objects and the same positions but the small bar is merged with its corresponding gear.



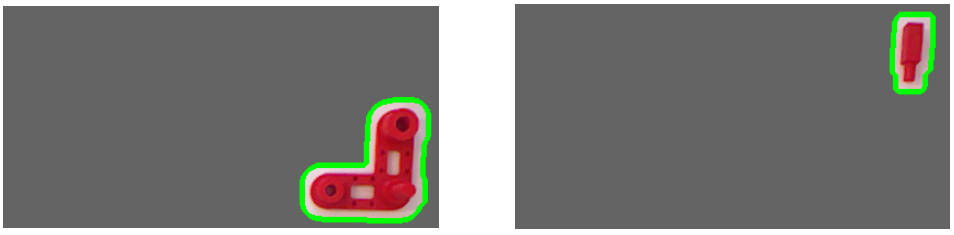
(a) first frame (b) second frame

Figure 11: Scenario one

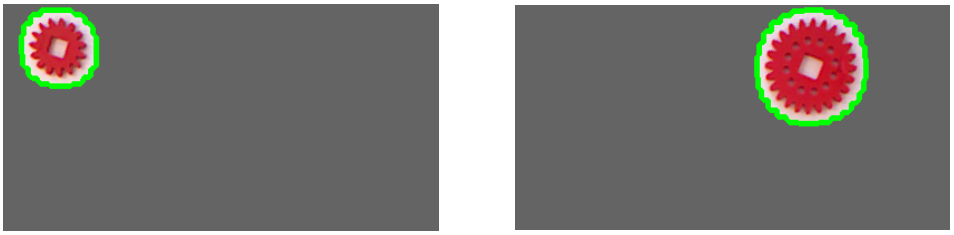
The detected objects with their numbering for first and second frames are represented on figure (12) and (13), respectively.



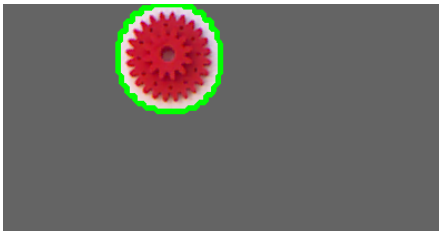
(a) Small Bar: Contour No. 1 (b) Hollow Gear: Contour No. 2



(c) L-shaped Support: Contour No. 3 (d) Bar: Contour No. 4

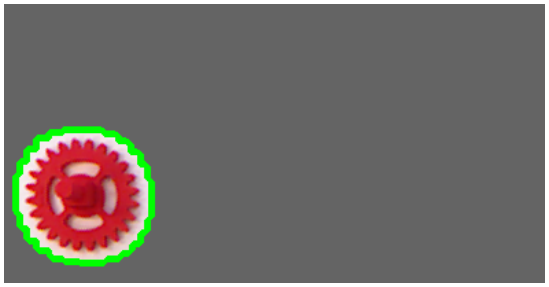


(e) Small Gear: Contour No. 5 (f) Large Gear: Contour No. 6



(g) Double-row Gear: Contour No. 7

Figure 12: Detected objects with their numbering by model_1 for input frame shown on figure (11a)

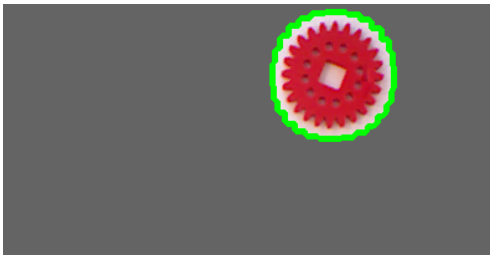


(a) Hollow Gear with Small Bar: Contour No. 1

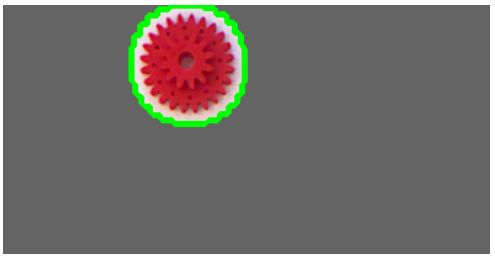
(b) L-shaped Support: Contour No. 2

(c) Bar: Contour No. 3

(d) Small Gear: Contour No. 4



(e) Large Gear: Contour No. 5



(f) Double-row Gear: Contour No. 6

Figure 13: Detected objects with their numbering by model_1 for input frame shown on figure (11b)

Corresponding Kullback-Leibler divergence measure between each couple of detected distributions of model_1 for the given input image and depth sensor data of Kinect is provided on table (3).

		Figure 13 Object					
		Hollow Gear with Small Bar	**L-shaped Support**	**Bar**	**Small Gear**	**Large Gear**	**Double-row Gear**
Figure 12 Object	**Small Bar**	28.6	36.49	92.05	71.84	49.05	47.14
	Hollow Gear	3.22	76.43	271.88	57.95	35.44	10.57
	L-shaped Support	30.09	0.03	14.65	50.71	7.43	24.31
	Bar	293.66	20.32	0.15	365.39	47.36	182.72
	Small Gear	30.1	212.66	217.31	0.83	94.19	20.86
	Large Gear	23.23	18.3	116.49	71.37	0.11	10.49
	Double-row Gear	9.77	40.02	75.25	12.25	10.5	0.09

Table 3: Kullback-Leibler divergence between detected objects from data corresponding to figure(11a) in columns and objects from data corresponding to figure(11b) in rows

In spite of highlighting the minimum value of each columns on table (3), each columns of this table from minimum value to the maximum value in ascending order is sorted, and their corresponding indices are logged at table (4). Therefore, the highlighted row of table (4) specifies the number of object on the frame (11a) which is the most similar one to the object corresponding to each column on figure (11b). So according to table (4), based on Kullback-Leibler divergence the objects number 1 to 6 on figure (13) are detected to have the least difference with objects 2, 3, 4, 5, 6 and 7 respectively on figure (12).

	Figure 13 Object					
	Hollow Gear with Small Bar	**L-shaped Support**	**Bar**	**Small Gear**	**Large Gear**	**Double-row Gear**
Figure 12 Object	Hollow Gear	L-shaped Support	Bar	Small Gear	Large Gear	Double-row Gear

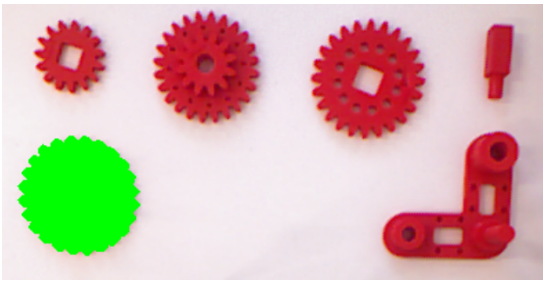
Table 4: Matched objects based on Kullback-Leibler divergence from data corresponding to figure(11a) and figure(11b) using model_1

Also, the divergence values corresponding to each cells of table (4) is provided on table (5).

	object number on figure (13)					
	Hollow Gear with Small Bar	**L-shaped Support**	**Bar**	**Small Gear**	**Large Gear**	**Double-row Gear**
divergence value	3.22	0.03	0.15	0.83	0.11	0.09
	9.77	18.3	14.65	12.25	7.43	10.49
	23.23	20.32	75.25	50.71	10.5	10.57
	28.6	36.49	92.05	57.95	35.44	20.86
	30.09	40.02	116.49	71.37	47.36	24.31
	30.1	76.43	217.31	71.84	49.05	47.14
	293.66	212.66	271.88	365.39	94.19	182.72

Table 5: Sorted Kullback-Leibler divergence between detected objects from data corresponding to figure(11a) in columns

According to table (5) and (4), it is concluded that among the detected pair of objects, the pair including the object number 1 of figure (11b) and object number 2 of figure (11a) which match with each other with Kullback-Leibler divergence equal to 3.22 has the maximum divergence among the detected pairs of objects. Therefore, this fact demonstrates that the missing object from figure (11b), which is object number 1 on figure (11a), has been merged with object number 2 in figure (11a). The detection result as a point cloud is also shown on figure (14).



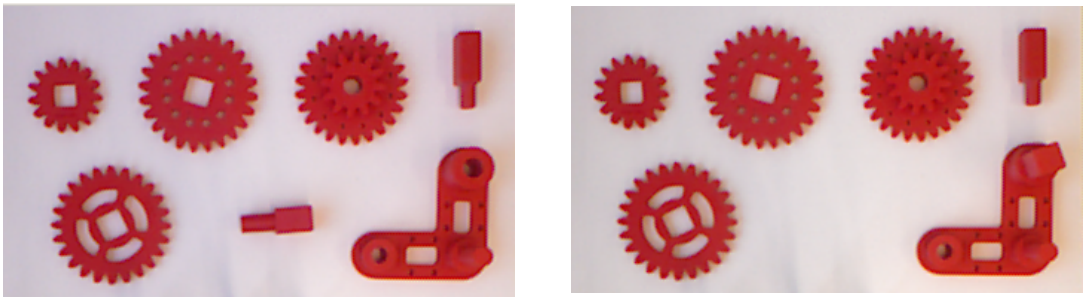
(a) Detected pair of merged objects

(b) Point cloud of detected pair of merged objects

Figure 14: Inference of merged components for scenario one

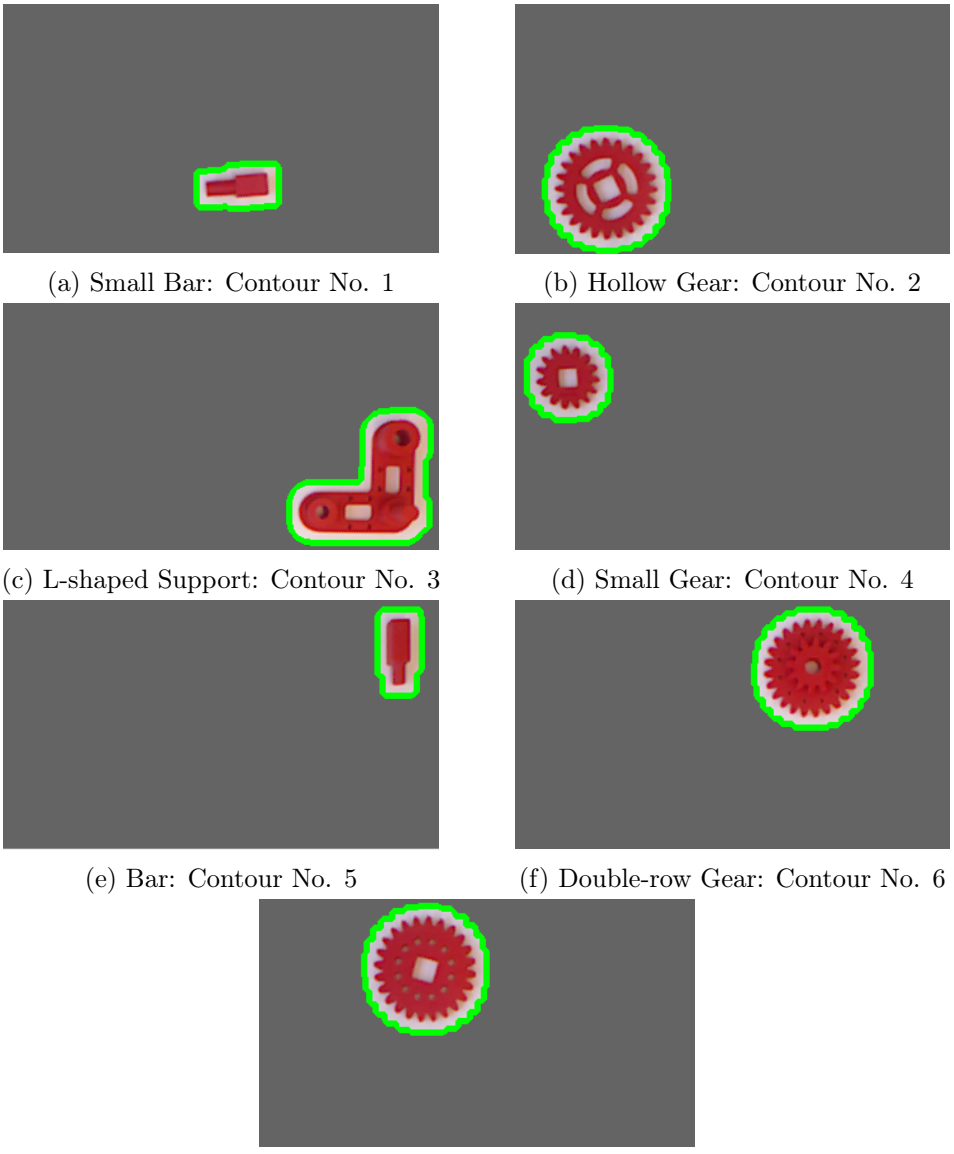
4.3 Scenario Two

Using the method discussed on previous section for *Scenario One* based on model_1 for examination of the results for assembly of each two pairs of figure (11a), successfully it can be inferred which object has been merged with which one except when the gear bar on figure 12(a) is assembled with L-shaped component on figure 12(c). Therefore, in this scenario detection of the merged objects is made to be able to detect also merging of these two components. First, the performance of model_1 for this scenario is examined. For this reason, RGB and depth sensor data corresponding to figures (15a) and (15b) are given to model_1.



(a) First frame (b) Second frame

Figure 15: Scenario two input frames



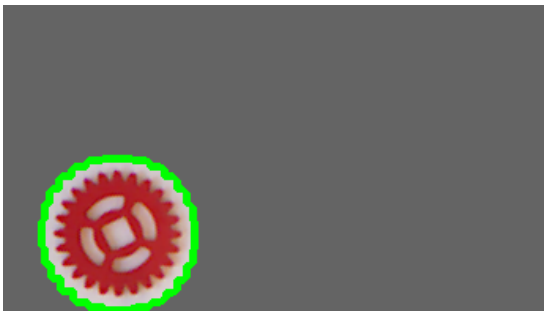
(a) Small Bar: Contour No. 1 (b) Hollow Gear: Contour No. 2

(c) L-shaped Support: Contour No. 3 (d) Small Gear: Contour No. 4

(e) Bar: Contour No. 5 (f) Double-row Gear: Contour No. 6

(g) Large Gear: Contour No. 7

Figure 16: Detected objects with their numbering by model_1 for input frame shown on figure (15a)

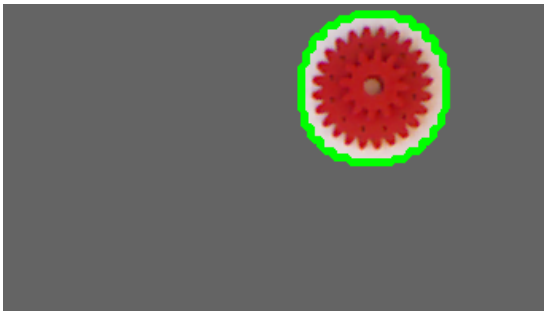


(a) Hollow Gear: Contour No. 1

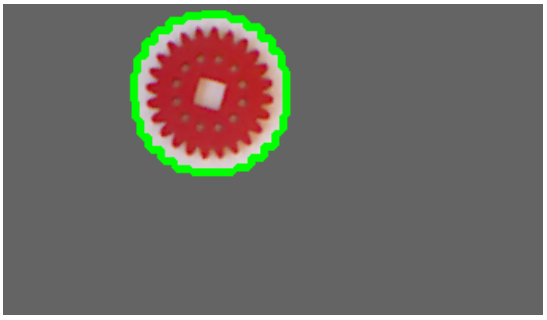
(b) L-shaped with Small Bar: Contour No. 2

(c) Small Gear: Contour No. 3

(d) Bar: Contour No. 4



(e) Double-row Gear: Contour No. 5



(f) Large Gear: Contour No. 6

Figure 17: Detected objects with their numbering by model_1 for input frame shown on figure (15b)

Finally, after computation of the Kullback-Leibler divergence between Gaussian distribution corresponding to each of the detected contours of figure (16) with each distribution of detected contours on figure (17) the results are logged on table (6)

		Figure 17 Object					
		Hollow Gear	**L-shaped with Small Bar**	**Small Gear**	**Bar**	**Double-row Gear**	**Large Gear**
Figure 16 Object	**Small Bar**	26.88	18.15	59.41	49.35	33.02	43.64
	Hollow Gear	0.04	57.11	53.59	378.24	24.52	2.9
	L-shaped Support	16.0	0.2	30.89	17.32	5.35	16.48
	Small Gear	30.57	112.18	0.08	234.28	51.13	25.7
	Bar	178.22	21.18	249.35	0.04	40.96	131.76
	Double-row Gear	13.32	11.36	23.53	61.83	0.01	5.29
	Large Gear	5.21	35.87	37.57	182.54	8.64	0.02

Table 6: Kullback-Leibler divergence between detected objects from data corresponding to figure(15a) in columns and objects from data corresponding to figure(15b) in rows

Minimum divergence value of each column of table (6) is highlighted which points to the matching pair of objects. Additionally, sorting the results shown on table (6) finally it is concluded that model_1 is capable of matching the objects between two frames but it is not able to decide which object has been merged with which one. In other words, among the detected pair of distributions based on analysis addressed at previous section, the maximum divergence between the distributions is expected to be for the object which also another object has assembled with it. Thus, in this scenario model_1 is not able to infer merged objects. One reason is the fact that there is noise at the captured data. In order to reduce the effect of noise, model_4 is provided.

Now the result of Kullback-Leibler divergence after sorting and matching the divergence values between each possible pair of objects from table (7) it is concluded that objects are detected correctly and the corresponding Kullback-Leibler divergence between detected pairs is logged on table (8).

	Figure 17 Object					
	Hollow Gear	**L-shaped with Small Bar**	**Small Gear**	**Bar**	**Double-row Gear**	**Large Gear**
Figure 16 Object	Hollow Gear	L-shaped Support	Small Gear	Bar	Double-row Gear	Large Gear

Table 7: Matched objects based on Kullback-Leibler divergence from data corresponding to figure(11a) and figure(11b) using model_1

	Figure 17 Object					
	Hollow Gear	**L-shaped with Small Bar**	**Small Gear**	**Bar**	**Double-row Gear**	**Large Gear**
Kullback-Leibler divergence	0.0271	0.0328	0.0149	0.0258	0.0024	0.0068

Table 8: Kullback-Leibler divergence value between distribution of each **matching** pair of objects in figure (16) and (17)

According to table (7) and (8), by assuming the fact that one object from first frame has been merged with another object and considering the maximum Kullback-divergence between distributions of matching pair of objects, it is concluded that the object number 1 of figure (16) has been merged with object number 3 of the same figure as it is shown as object number 2 in figure (17).

4.4 Data Noise Reduction

One of the sources of having unsuccessful results utilizing model_1 is noise in input data. So, it is necessary to reduce the influence of input data noise in order to be able to increase inference capability of the system which allows to detect merged objects. Now a comparison between model_4 and model_1 is studied in order to demonstrate noise reduction capability of model_4. Two input frames taken from an identical scene shown on figure (15a) are provided to model_1 and model_4. The final results are represented on table (9).

	object name on figure 16						
	Small Bar	**Hollow Gear**	**L-shaped Support**	**Small Gear**	**Bar**	**Double-row Gear**	**Large Gear**
model_1	0.007	0.044	0.006	0.072	0.034	0.011	0.017
model_4	0.004	0.03	0.003	0.002	0.11	0.006	0.002

Table 9: Kullback-Leibler divergence results between detected pair of objects from data corresponding to figure (16) based on results of model_1 and model_4

Ideally, Kullback-Leibler divergence between pair of detected pair of objects should be equal to zero inasmuch as the each pair represent one identical object therefore the normal distribution fitted to each object at each pair should be identical. However, there is nose in data acquisition which influences the final results so the divergence between each pair is not equal to zero. According to table (9), utilizing model_4 decreases the divergence between each detected pair of objects except for object number five on figure (16e). This happens because of the fact that model_4 downsamples data which eliminates the furthest data points on the point cloud, and as object number five is a small gear bar on the top right corner of the taken frame, less number of reliable data-points remains for clustering so it influences the final result.

5 Conclusion

At the beginning, different unsupervised learning algorithms including GMM, Birch, Mean-Shift, K-means and DBSCAN were used to cluster the image and depth sensor data. However, it was not possible to fit the clusters on each objects quite separately in as much as data are too noisy and the components are too small and near each other. Afterwards in this project, the capability of inferring small assembly objects is demonstrated by combining the machine learning unsupervised learning with computer vision algorithms. It is proved that vision techniques can assign a contour around each object. Next, the object type is inferred by fitting a Gaussian model to each detected contours. Then, the noise effect on the results decreased considerably by utilizing image registration. Finally after reducing the noise effect, it was possible to infer which pair of the objects merged together. Nonetheless, the hyper-parameters of the final proposed algorithm require to be tuned for any new scenario; These hyper-parameters include threshold for removal of the contours detected for the holes inside the hollowed gear, the voxel size for down-sampling, the ratio of maximum distance between point clouds used for noise reduction, and the point numbers for KNN[1] used for FPFH feature calculation.

6 Appendix

6.1 GitHub Repository

IAOD[2] GitHub repository was created for version control of the project and systematic improvement and record of codes. This allowed the author to facilitate testing different possible methods by logging the changes in a clear way. Herewith HTTPS and SSH links to the repository are provided:
https://github.com/mahdinobar/IAOD.git
git@github.com:mahdinobar/IAOD.git

7 References

References

[1] Contour approximation method. [Online]. Available: https://docs.opencv.org/3.4.2/d4/d73/tutorial_py_contours_begin.html

[2] Fast point feature histograms (fpfh) descriptors. [Online]. Available: http://pointclouds.org/documentation/tutorials/fpfh_estimation.php

[3] Function to compute fpfh feature for a point cloud. [Online]. Available: http://www.open3d.org/docs/python_api/open3d.registration.compute_fpfh_feature.html

[4] Gaussian mixture models. [Online]. Available: https://scikit-learn.org/stable/modules/mixture.html#mixture

[1] K-Nearest Neighbors
[2] Inferring Assembly Objects from Demonstration

[5] Global registration. [Online]. Available: http://www.open3d.org/docs/tutorial/Advanced/global_registration.html

[6] Icp registration. [Online]. Available: http://www.open3d.org/docs/tutorial/Basic/icp_registration.html

[7] Image registration. [Online]. Available: https://en.wikipedia.org/wiki/Image_registration

[8] Innovation challenge. [Online]. Available: https://new.siemens.com/us/en/company/fairs-events/robot-learning.html

[9] Introducing scikit-learn. [Online]. Available: https://jakevdp.github.io/PythonDataScienceHandbook/05.02-introducing-scikit-learn.html

[10] Kinect for xbox 360 and kinect for windows (kfw) v1 specs. [Online]. Available: https://zoomicon.wordpress.com/2015/07/28/kinect-for-xbox-360-and-kinect-for-windows-kfw-v1-specs/

[11] Kullback–leibler divergence. [Online]. Available: https://en.wikipedia.org/wiki/Kullback%E2%80%93Leibler_divergence

[12] Open source computer vision. [Online]. Available: https://en.wikipedia.org/wiki/OpenCV#Applications

[13] The pcl registration api. [Online]. Available: http://pointclouds.org/documentation/tutorials/registration_api.php

[14] Point feature histograms (pfh) descriptors. [Online]. Available: http://pointclouds.org/documentation/tutorials/pfh_estimation.php#pfh-estimation

[15] Preprocessing data: Standardization, or mean removal and variance scaling. [Online]. Available: https://scikit-learn.org/stable/modules/preprocessing.html

[16] Vision guided robotic systems. [Online]. Available: https://en.wikipedia.org/wiki/Vision_Guided_Robotic_Systems

[17] What is camera calibration? [Online]. Available: https://www.mathworks.com/help/vision/ug/camera-calibration.html

[18] J. Heikkila and O. Silven., "A four-step camera calibration procedure with implicit image correction," *International Conference on Computer Vision and Pattern Recognition*, vol. 22, 1997.

[19] S. Suzuki, "Topological structural analysis of digitized binary images by border following," *Computer Vision, Graphics, and Image Processing*, 1985.

[20] Z. Zhang, "A flexible new technique for camera calibration," *Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, 2000.

[21] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv:1801.09847*, 2018.