Berlin University of Technology

A thesis submitted for the degree of
Bachelor of Science

Department of Computer Science and Electrical Engineering

Three dimensional Joint Detection

Author:
Andreas Orthey

Supervisor:
Prof. Oliver Brock
Second Supervisor:
Prof. Marc Toussaint
Thesis Advisor:
Dov Katz

December, 2010

Contents

List of Figures

List of Tables

List of Algorithms

1 Abstract

Kinematic structures are currently one of the most important prerequisites for robot technology. Grasping objects, painting cars or performing household tasks are just a few examples from the manifold space of possible applications. To execute such operations, the structure of an object has to be known a priori. In dynamical changing environments, however, this prior specification becomes unfeasible. Therefore, an automated process is required that can reliably extract kinematic structures from visual sensory input. This work contributes to this effort by acquiring joint types from moving 3d point clouds. A cloud consists hereby of presegmented visual features from object parts or the environment. The algorithm tries to classify each relationship in the point cloud according to different categories of joint types. Several real world experiments with ordinary objects like doors, drawers, tricycles, and laptops were performed, to test the stability of this approach.

Zusammenfassung

Kinematische Strukturen sind zurzeit eine der wichtigsten Grundlagen der Robotik. Sie ermöglichen es, unter anderem, Objekte zu greifen, Autos zu lackieren oder einfache alltägliche Aufgaben zu erledigen. Um solche Tätigkeiten durchzuführen, ist jedoch Vorwissen über die Struktur der jeweils involvierten Objekte verlangt. In Umgebungen, die sich dynamisch verändern, ist es jedoch nicht möglich solch ein Vorwissen bereitzustellen. Deshalb wird ein automatischer Prozess benötigt, welcher kinematische Strukturen aus visuellen Informationen extrahieren kann. Diese Arbeit unterstützt diese Bemühungen durch die Extraktion von kinematischen Strukturen, aus sich bewegenden 3d Punktwolken. Eine Punktwolke besteht aus bereits vorsegmentierten visuellen Features von Objekten oder der Umgebung. Der Algorithmus versucht jede Beziehung innerhalb der segmentierten Features anhand von verschiedenen Gelenk Kategorien zu klassifizieren. Um die Stabilität des Ansatzes zu überprüfen, wurden mehere Experimente in der realen Welt durchgeführt. Dabei wurden alltägliche Gegenstände wie Türen, Schubladen, Dreiräder und Laptops benutzt.

2 Introduction

2.1 Motivation

In the last decades, robot technology achieved several useful accomplishments to improve, simplify and automate parts of human life. Examples range from automatic processes in car manufactures and household tasks to repairing duties at the international space station. The underlying framework, which enables most of these successes, is based on so called kinematic structures. Those are models, which describe the motion of a manipulator or objects to interact with. Before a robot can start its task, a human operator has to specify those models apriori. However, this can become a very time consuming task, constrains the autonomy of the machine, and will be unfeasible in dynamical changing environments.
Therefore, it seems to be necessary to automate this process. The robot should be able to acquire the structure of an arbitrary object, without prior knowledge or intervention of a human operator. Once the structure is obtained, further goal directed manipulation will be possible. Especially in human unfriendly environments, like nuclear waste plants or on other planets, this seems to be a desirable skill.
While there are many attempts to solve this problem, there is no general solution. One possible reason is the concentration of vision research on still images. While an object is at rest, the underlying structure can only be guessed. But once the object moves, the inherent properties can be seen and in principle be analyzed. In this context, this thesis will contribute to a larger project with the emphasis on learning intelligent interactions in order to reveal structures from moving objects.
To reach this goal, a module has to be designed, which can reason about structures, given only observations over time. An observation consists hereby of presegmented 3d point clouds of a moving rigid object. Given this input, the developed algorithm concentrates on explaining the motion in terms of different joint types. This is useful for further applications, like learning the right interaction to reveal the structure. In the next section 2.2 a brief explanation of the developed methods is given.

2.2 Contributions

The main purpose of this approach is to extract a simplified kinematic structure from moving rigid objects, described by presegmented 3d data points. Kinematic structures can be described in terms of rigid body parts, or links, which are connected through joints. A complete description involves the knowledge about the relationships between each two links. Therefore, this thesis concentrates on revealing the type of joint between all segmented clusters of the 3d data points.
The main problem in this context is to find a way of obtaining the relative motion between two links. It is a lot easier for example to explain the motion of a door and its surrounding wall, because one of the objects does not move. But extracting the relative motion between a car and its wheels is more difficult, because they move together. How to obtain a solution will be the main focus of chapter 4.4.
After the relative motion is revealed, expert-designed joint types will be used to char-

acterise the relation. Depending on the used methods to obtain the 3D data points, the dataset can consist of noise. Therefore, it is essential to design joint types, which are able handle a suitable amount of noise. The methods will be described in section 5, where also limitations of this approach will be discussed.
Another crucial problem, which arises, if the dataset is obtained from just one camera or without depth information, is scale ambiguity. Since the size of an object unknown without prior experience, it can be at an indefinite number of distances relative to the camera. Therefore, if it is compared to another object with different real depth, an additional prismatic motion occurs. But at the right scale, this prismatic motion disappears. This is why the problem can be rewritten as an optimization problem. If a scale factor can be found, where no prismatic motion occurs, it is likely, that this motion is introduced by scale. A detailed discourse of this topic can be found in section 5.5. Before discussing the methods, the upcoming section 3 will provide a background to kinematic structures, introducing related work and briefly summarising a possible acquisition process of the input data. Additionally, a more technical discourse about the main scope of this thesis will be provided in section 3.4.

3 On detecting kinematic structures

3.1 Background

As described in [3], kinematics is the science of motion, which is concerned with the geometrical and time-based properties of an object. Forces and torques are thereby neglected, which means, that the main focus lies on the pure movement, not the inherent dynamics. To describe objects, kinematic structures are used consisting of links and joints. Links are rigid bodies or body parts where the distance of two arbitrary points is always constant. Each link can be connected to one another or to the environment through joints. In general, the relation between two links can be described by a total of six parameters, three positions and three orientations. The number of parameters to describe the motion is referred to as degrees of freedom. Joints will constrain the motion between two links by reducing their degrees of freedom. If you take a look for example on a revolute joint present in a door, the motion can be described by one rotational component only, which is why the degrees of freedom are reduced to one. The second single degree of freedom joint is called prismatic. This one can be described by a translational parameter and can be found for example in a drawer. Those two are the most common joint types in everyday objects. Examples are bicycles, cars, scissors, laptops and several others. Objects which exposure this structure are referred to as articulated.

3.2 Related Work

There are already several published works on identifying kinematic chains from motions and observations. Most of them provide solutions to simplified settings.
Kirk *et al.*[11] observed the motion of a human to infer a skeletal information. Hereby he and his colleagues assumed, that each joint can be represented by a revolute joint. Input data is obtained from a motion capturing device, with several markers attached to the human body. To identify rigid body parts, they calculated the standard deviation in distance between all marker pairs. The relationships between body parts were formulated by them as a nonlinear optimization problem, where the revolute joint position is located at the line, where the distance between all markers is minimal. Because a revolute joint can be seen as attached to both bodies, the distance at the axis to each point on the bodies should remain the same. One weakness of their approach is the usage of revolute joints only. This constraints the number of objects, which can be analyzed. If one seeks for autonomous behaviour, attaching artificial marker also does not seem to provide the right direction. A general framework should also handle informations, coming from monocular camera perspectives.
A more general approach, but still relying on artificial markers, is proposed by Sturm *et al.*[20, 21] They analyzed the kinematic structure in open kinematic chains, that means objects, where at least one of the parts is not in motion[1]. Their goal is to fit

[1]A closed kinematic chain consists of arbitrary moving links, which are not static or connected to the environment. Open kinematic chains on the other side always have one non-moving part, which is connected to the environment. One example is a robotic manipulator, which is rigidly connected to

a set of models to the observed motion between each pair of object parts. Each part is considered as a node in a graph, where the edges between them represent the type of connection. A possible arrangement of the object parts is represented by a spanning tree. Because there are many such trees, the one, which maximizes the expected likelihood of a new observation is choosen to represent the kinematic structure. In every step, when a new observation arrives, the model is updated. For enough training samples, the algorithm converges to the right structure. This has also applications in system identification processes [22]. While this sounds promising, it still can only deal with open kinematic structures. But in a real world setting, there are many objects, which are moving independently and do not have non moving parts. The simplest example is a scissor, where both parts are moving in relation to each other.
In terms of input, a more realistic setting is proposed by Dearden and Demiris [4]. They abandon artificial markers, but use image features, which they track over time. Features, which move together are so called optical flow points and clustered into one object. Instead of extracting the kinematic relationship, the main goal is to imitate the observed motion. One example is the movement of a clipper tool, which could imitate the movement of two hands in front of the camera. Nevertheless, their approach was only showed in a very simplistic setting and not with a wide variety of objects.
Taycher, Fisher III and Darell [24] operated on orthographically projections of 3d objects. From this 2d projection, they tried to recover the articulated structure of the object. In their experiments, they used simulated data with added noise to demonstrate the robustness. Because they relied on a very small amount of gaussian noise (1 pixel standard deviation), it is arguable, if this can be used with real world data. Another critical point is the assumption, that object parts are structured as a kinematic tree. Every chain-like object would be rejected.
Another related approach to this work is conducted by Pollefeys et al. [27]. They used image features on objects and tried to infer the kinematic relationship by means of affine projections. That means, they do not use 3d informations but rely on the monocular camera image. In their work, they introduce motion subspaces. That are basically classes of features which move together and belong therefore to one space. If a joint between two motion subspaces is present, than there is a position, where the distance to each feature in both motion subspaces is constant. This holds true for revolute joints. A feature, which rotates about the axis of rotation, has a constant distance to this axis. While their approach is only valid for revolute joints, it can handle non rigid body parts like a face. Handling non rigid body parts is currently not a part of this thesis. Therefore, their methods could act as one of further improvements.
Katz et al. [8, 9] tracked features in 2d and constructed a graphical representation from them. This graph consisted of vertices, representing the features, and an edges between features, if the distance remained constant. To extract clusters and therefore the links of the underlying object, they grouped the graph into parts of high connectivity. Once the links are extracted, their algorithm searches for a transformation between pairs of links, which can best explain the motion. A transformation is basically a pure translation, if a prismatic joint is present between two links. On the other side, a

the ground.

revolute joint consists of a pure rotational component. If the transformation cannot be described by one of the mentioned transformations, it is declared as disconnected. Because only 2d planar objects are observed, a huge number of real world objects are discarded. While scissors on a table can be analyzed, there methods will already fail for doors.
Ross, Tarlow and Zemel [18] use 2d projections of 3d observations to infer a kinematic structure. First they group the 2d features into rigid sticks. Each stick represents a rigid body part. While observing an articulated object, sticks are not independent, but the ends are more likely connected to each other. Therefore, they use an Expectation-Maximation (EM) algorithm to learn the correct position of the sticks from newly observed data. At each new iteration the algorithm will be further improved, till it will converge to the right structure. While their assumption of connected sticks is valid for humans, it will likely fail for rigid bodies, which are not connected at the end but in the middle, like scissors.
All papers discussed so far use either artificial markers, can only deal with 2d informations or lack the ability to handle closed kinematic chains. In contrary, the proposed algorithm in this work does not rely on artificial markers only. Only two assumptions are used for the input data. First, it consists of 3d points on the objects which have to be identified. Second, the points are already grouped into different clusters according to different parts of the analyzed object. Given this information, the proposed method can handle noisy input and is especially able to identify structures from closed kinematic chains. One way of acquiring the input information will briefly discussed in section 3.3.

3.3 3D Image features

Before analysing the structure of the main work, a brief summary about one possibility to obtain the sources of the input data will be given. To model the kinematic structure of an object, the first step is to identify interesting points. This can be achieved with artificial markers, like the motion tracking devices discussed in the last section. But this would assume, that the environment is already structured. Someone has to add markers to the object. This raises the question, if in this case, it isn't easier to model the whole object by a human operator. If artificial markers are necessary, the robot is not autonomous anymore and still relies on a helping hand. Especially in environments like outter space or at nuclear waste factories, this cannot be an option.
Katz et al. [10] on the other side acquire the 3d informations from feature tracking only. There approach is already robust against different lightning conditions and shows how the input data can be produced without prior knowledge. While it is a reliably source of 3d features it still got two problems. First, they acquired the features through a monocular non-moving camera. This is why it lacks a proper depth information. Produced features could be produced by the same object but greater and further away from the camera. To generalize the work, this scaling ambiguity will be handled by the algorithm. Second, they used a low resolution camera. As one could imagine, the degree of noise is slightly increased by this setting.
Especially because its non perfect acquisition process, the data set provides a very

good testing framework. If the algorithm works under this conditions, it will also work with more carefully conducted experiments. The results will be discussed in section 6, after the work is explained in detail. For the upcoming parts of this thesis, the exact method of obtaining the features will be neglected. The knowledge about the possible incoming data shows, that a highly generalized method is desired.

3.4 Joint Detection

The formal input to the algorithm are N clusters of image features, which represent different parts of a body or interesting regions in the surrounding environment. If the type of joint between different object parts is known, the kinematic structure is easier inferred. This is why the problem will be simplified, by picking each two pairs of clusters in the scene, and analyze its motion in terms of joints separately. Additionally to the type of joint, also a confidence value is necessary to show how sure the algorithm is about a certain type. Putting everything together results in an structure sketched in algorithm 1. It states that the goal is to determine the type of joint and a confidence value from the input cluster. The *JointDetection* function can be seen as the algorithm this thesis is about. It will be examined in every detail in the next sections. By taking just two clusters at a time, the overall problem is divided into $\binom{N}{2}$ smaller problems which can easier be solved. Moreover, each of the pairs can be calculated in parallel if more processors or parallel computing architectures are avaiable.

The subdivided problem of determing the joint between two clusters becomes now more handy. But before the relationship between them can be determined, the relative motion has to be obtained. What does this mean? Consider two objects like a wall and a door. To acquire the relative motion between the door and its surroundings becomes easy, because the wall does not move with respect to the camera. Therefore, the relative motion is equal to the motion of the door. But if one considers, that the second object is also floating through space arbitrarily, the former easy appearing task will become very hard to solve. The complete section 4.2 is dedicated to explain this problem in more detail and obtain a robust solution.
As already covered in section 3.3, there are many concepts involved in generating the input. Every step is prone to uncertainty and could in the worst case consist of a huge

Algorithm 1 AnalyseKinematicStructure

```
Require: $O_{i=1}^{N}$
Ensure: $J_{i=1}^{\binom{N}{2}}, P_{i=1}^{\binom{N}{2}}$
1: $m \leftarrow 0$
2: for each pair in $O_{i=1}^{N}$ do
3:     $J(m), P(m) \leftarrow JointDetection(pair_m(O_{i=1}^{N}))$
4:     $m \leftarrow m + 1$
5: end for
```

amount of noise. Methods, which will reliably detect joints, apparently have to deal with uncertainty. This aspect will also be discussed during the development of the methods in section 5.
In section 6 the algorithms performance on ordinary objects like doors, drawers, or tricycles will be shown. Also a brief description of limitations will be discussed.
At the end in section 7, a conclusion about the whole algorithm will be presented. There, the question is answered, if the overall problem of detecting kinematic structures from observations is hereby completely solved.

4 Relative motion

4.1 Terminology

In order to discuss the two core problems of revealing the local motion and determing the relationship between two bodies, a list of formal definitions is given.

- **Point:** A coordinate in 3d space represented by x,y,z.
- **Body:** A set of points representing a rigid body. That means, all points are located somewhere on this body.
- **Feature:** A point on a body. If the body moves, the feature still stays the same, but its x,y,z coordinates change over time.
- **Trajectory:** The time sequence of one feature. That are all possible x,y,z values, which are observed for a particular feature.
- **Timeframe:** The position of a set of features on a body, at a specific sequence in time.
- $(\mathbf{B}_{\boldsymbol{\Lambda}})_{\mathbf{i=1}}^{\mathbf{M}}$: A Body labelled Λ, with information about the position of its points at M different timeframes. This can be seen as equal to all features, representing body Λ.
- $\mathbf{B}_{\boldsymbol{\Lambda}}(\mathbf{i})$**:** The position of all points on a body at a given timeframe i.
- **H:** A matrix H represents a homogenous 4x4 matrix, associated to a rotation matrix R and a translation vector T.
- **p:** A probability. Is used as a score, for example to determine how good a model has matched the observations. Also used as confidence score for a certain type of joint.

4.2 Problem description

If two bodies are moving arbitrarily through space, it is hard to determine the relationship between them.
Take a look at the motion of a tricycle in figure 1. Just using feature trajectories on the wheel (blue) and the top (black) of a tricycle do not act as good indicators of its true motion. The knowledge of the real relationship is shadowed to the observer. A mechanism is necessary, which can hold one of the body parts at rest. If one just observes the relative motion (red), it becomes obvious, that a revolute joint is present between wheel and top.

To obtain such a result, the harder problem of revealing the motion between moving bodies is projected onto the easier problem, where one of the bodies does not move at all. If one of the bodies is fixed, a prismatic or revolute motion is clearly visible as a straight line or a circle, respectively. Therefore, to reveal the local motion between

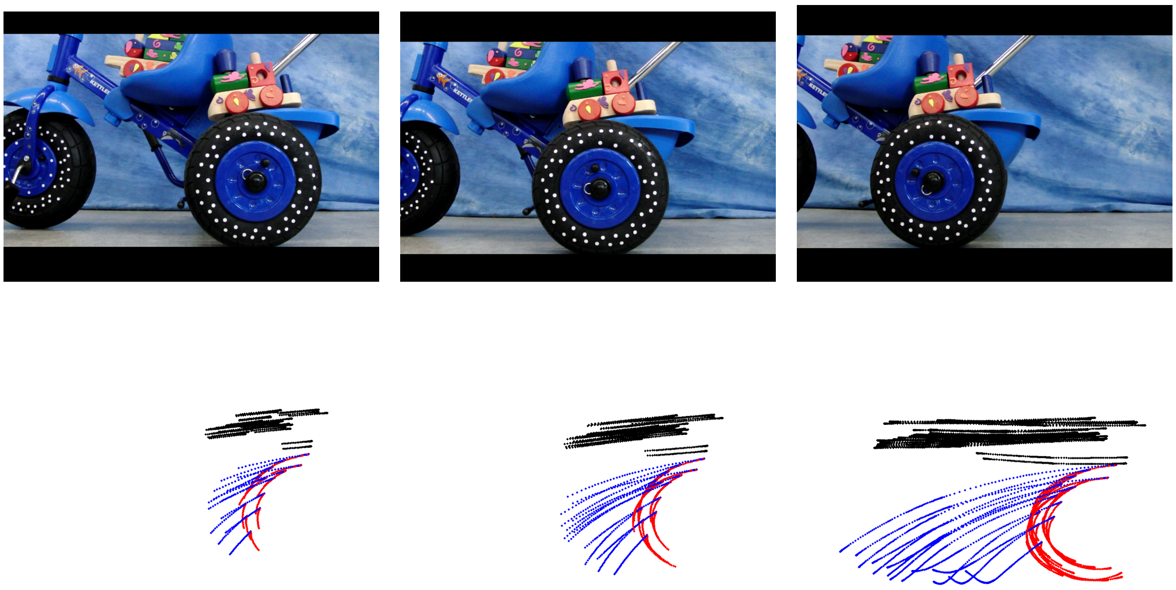


Figure 1: Motion of a tricycle: The wheel over time (blue), the top of the tricycle (black) and the relative motion between wheel and top, obtained from the algorithm (red)

two bodies, the first step is to fixate one of them. This is basically done, by choosing one of the two objects as a reference. In each time frame, the transformation of the reference body is calculated, between its current and its previous timeframe. If this is known, the body can be projected back and fixed to its previous position.
As it will be shown in the upcoming sections, if the transformations between all time frames of the reference body are known, the relative motion with respect to another body can be calculated by use of homogenous transformations only.
In section 4.3 the basics of calculating a homogenous transformation between two point clouds are discussed. That means, the algorithm tries to obtain a rotation and translation, which best transforms one point cloud into another one. This problem is commonly referred to as point cloud to point cloud registration. Several different methods, mostly iterated solutions, are discussed in [13]. But in the case, where both clouds have the same amount of points, a robust closed form solution can be obtained [6, 1].
Once the best transformation between two clouds is found, section 4.4 will propose an algorithm to obtain the relative motion. To clarify the details, at first, the algorithm is explained in the 2d-case with just two time frames. By understanding this, it will become clear, how this method can be applied to the general 3d case. At the end, a robust and easy to implement solution will be achieved. It is tested on simulated data and also with real world objects from challenging data sets (see section 6).

4.3 Point Cloud to Point Cloud Registration

Each input body is represented by a set of points. If a body moves, there is a transformation, which maps the old point cloud onto the new one. This can be represented by a homogenous transformation matrix HT, which is basically a $4x4$ matrix, consisting of a rotation R and a translation T. A direct method, which computes the homogenous transformation in the present of noise is proposed by [1]. It assumes that there are at least six points and that both point clouds aren't an exact match.
Let $P = p_1, ..., p_N$ and $Q = q_1, ..., q_N$ denote two point clouds where $N \geq 6$. To find the homogenous transformation between them consists of finding R and T, which can transform one point cloud into another one. To achieve this goal, a least-square error function is used, which defines the error between the transformed point cloud and the final cloud. The solution to this problem will therefore lead to a minimization problem. The error can be defined as the squared sum of distances between the transformed P and the original Q.

$$E = \sum_{i=1}^{N} \|q_i - (Rp_i - T)\|^2 \tag{1}$$

The trick to solve this equation is to decouple the translation and rotation. In [7] it is shown, that the centroid of a point cloud remains the same, if one rotates the cloud about this point. This becomes clear, if one recalls, that the distance between each point to its axis of rotation remains constant. Therefore, the centroids of both point clouds are calculated and aligned to the origin of the current coordinate frame. Afterwards the task is reduced to find a rotation matrix, which aligns both shifted point clouds. To be more concrete, if p_c is the centroid of P, and q_c the equivalent of Q, the shifted point clouds are calculated by

$$p'_i = p_i - p_c \tag{2}$$

$$q'_i = q_i - q_c \tag{3}$$

Now, a new error function can be stated, which defines the alignment of both shifted clouds:

$$E = \sum_{i=1}^{N} \|q'_i - Rp'_i\|^2 \tag{4}$$

If a rotation R is found, which minimizes this error function, the translation vector of the original problem can be found via $T = q_c - Rp_c$.
To find R, the previous error function from (4) is rewritten as

$$\begin{aligned} E &= \sum_{i=1}^{N} \|q'_i - Rp'_i\|^2 && (5) \\ &= \sum_{i=1}^{N} (q'_i - Rp'_i)^T (q'_i - Rp'_i) && (6) \\ &= \sum_{i=1}^{N} (q'^T_i q'_i + p'^T_i p'_i - 2q'^T_i Rp_i) && (7) \end{aligned}$$

As can be seen from 7, in order to minimize the error function one has to maximize the expression $2q_i'^T R p_i$. This leads to a new error function.

$$\begin{aligned} E' &= \sum_{i=1}^{N} q_i'^T R p_i' && (8) \\ &= Trace(R \sum_{i=1}^{N} q_i' p_i'^T) && (9) \\ &= Trace(RH) && (10) \end{aligned}$$

In (10) the covariance matrix H is used, which is defined as $H = \sum_{i=1}^{N} q_i' p_i'^T$. To maximize the quantity RH a singular value decomposition (SVD) [16] is introduced. This decomposes H into $H = U\Lambda V^T$. In [1] it was shown, that if R is set to be $R = VU^T$, this will lead to

$$RH = VU^T U \Lambda V^T = V \Lambda V^T \quad (11)$$

and maximizes the initial term $Trace(RH)$. This is why the homogenous transformation matrix can directly be written as

$$\begin{pmatrix} R & T \\ \vec{0} & 1 \end{pmatrix} = \begin{pmatrix} VU^T & q_c - (VU^T)p_c \\ \vec{0} & 1 \end{pmatrix} \quad (12)$$

where q_c is the centroid of the point cloud Q, p_c the centroid of P, V and U^T the results from the singular value decomposition of the covariance matrix H.
So far there was no discussion about the completeness, that means, if this approach always finds a rotation matrix R. If the points in P are *not* coplanar, a unique solution can be found. The determinant of this rotation matrix is equal to one. But in the case, where all the points in P are coplanar, one can still find a unique rotation which maximizes the term RH, but also a a unique reflection which also leads to a valid solution. In this case, the determinant of R will have a value of minus one. To handle this case the eigenvalues of the covariance matrix H are evaluated. If one of them is equal or near to zero, the points are coplanar. This is equivalent to the result, that one of eigenvalues has zero length. This is why the sign can just be changed in this column of V to obtain the right rotation matrix. The same holds true, if the points in P are colinear, with the difference, that there are two zero values. But if there are huge outliers, this will effect the eigenvalues in a way that no correct R can be found. Therefore, it is sufficient to evaluate, if one of the eigenvalues is near zero. Otherwise, no correct solution can be obtained.
Algorithm 2 shows the complete function of this procedure. There are five additional methods used to calculate the homogenous transformation matrix. First, the *centroid* function calculates the mean value of each component of the input point cloud. Second, the *shift* method is used, which subtracts the centroid from each point cloud and aligns them so that the centroid matches the origin. Next, there is the *svd* function which performs a singular value decomposition, where the input matrix is decomposed in $M = U\Sigma V^T$, where U and V are unitary matrices and Σ is a diagonal matrix which

consists of the singular values [16]. The method *zeroValueExist* checks if one of the singular values of the covariance matrix is zero. Finally, *findZeroValue* returns the index of the singular value which is zero.
The output from the algorithm is a rotation matrix R and a translation vector T which can be converted into a homogenous transformation matrix as shown in (12). The homogenous transformation is a very general concept to explain the relation between two clouds. In the following section, it will be assumed, that the transformation is always known. As already seen, this is not the case, if there is too much noise or the data set consists of huge outliers. But if this is the case, already the clustering of the points is wrong and the whole method will likely fail. Therefore, it is sufficient to announce this failing and stop processing.

Algorithm 2 computeHomogenousTransformation

```
Require: $P_{i=1}^N, Q_{i=1}^N$
Ensure: $R, T$

 1: $q_c \leftarrow centroid(P)$
 2: $p_c \leftarrow centroid(Q)$
 3: $P_{i=1}^N \leftarrow shift(P, p_c)$// shift the point clouds to the origin
 4: $Q_{i=1}^N \leftarrow shift(Q, q_c)$
 5: $H \leftarrow \sum_{i=1}^N q_i p_i^T$
 6: $U\Lambda V^T \leftarrow svd(H)$
 7: $R \leftarrow VU^T$
 8: $T \leftarrow q_c - Rp_c$
 9: if $det(R) \neq 1$ then
10:    if $zeroValueExist(H)$ then
11:       $Index \leftarrow findZeroValue(H)$
12:       $V(Index,:) = -V(Index,:)$
13:       $R = VU^T$
14:       $T = q_c - Rp_c$
15:       return $R, T$
16:    else
17:       print ‘‘No solution’’
18:    end if
19: else
20:    return $R, T$
21: end if
```

4.4 Relative Motion

Based on algorithm 2, one can start finding the relative motion between two arbitrary moving bodies. Algorithm 3 provides an overview in pseudocode about the developed methods. It will be discussed in detail in this section.
A body is now defined by 3d image features which are moving over M timeframes. The goal is to acquire informations about the relative motion between each pairs of bodies in the scene. To start, one of the bodies is choosen as a reference body. Everytime this body moves, the transformation is calculated and it is aligned back to its first frame. The same is done for the second body. That means, if both of them are moving in the same way, this procedure will remove the same motion and will end up with two non-moving bodies.
The interesting part occurs, if the second body starts moving in addition to the first one. By removing the motion of the first body from the second one, the local motion between them will be revealed. If this is done, this motion can be examined with methods described in the next chapters.
Once the removing procedure of motion of the first body is understood, the algorithm becomes quite simple. For convenience, the motion between two consecutive timeframes of the first body is called the global motion. Corresponding to this global motion there is a homogenous transformation matrix H_G which relates both timeframes to each other. Further there is the local body motion, that is the position of the second body, if global motion is removed. This will be denoted by B_{Local}. For each consecutive timeframe of the local body there is again a homogenous transformation matrix H_L. To complete the declarations of the algorithm the last matrix is called $H_{complete}$, which relates the current timeframe to timeframe zero. This will be used to remove the global motion from each time step of the second body.
The first step towards revealing the local motion is to loop over all timeframes. In each step the relationship between the current timeframe and the following one is examined. To clarify this step, consider as an example two bodies at timeframes t_0 and t_1 (figure 2). The homogenous transformation between the first body H_G is computed by using the method described in the last section. This is the global motion, which is afterwards removed from the second body. Then, a transformation H_L between B_1 and the predicted position of body B_1 is computed. Together with H_G this transfers body B_0 to body B_1 by considering the global motion. Because H_G is known, the inverse will transfer B_1 back into the old time frame t_0, but with the additional transformation H_L. That means there is now an additional body B_1^{Local}, which is the old body B_0 transfered with the previous unknown transformation H_L. The homogenous transformation matrix is not only valid in such a simple 2d setting but is naturally defined for general 3d coordinates. That means, that this process is valid for 3d features to reveal the local body motion.

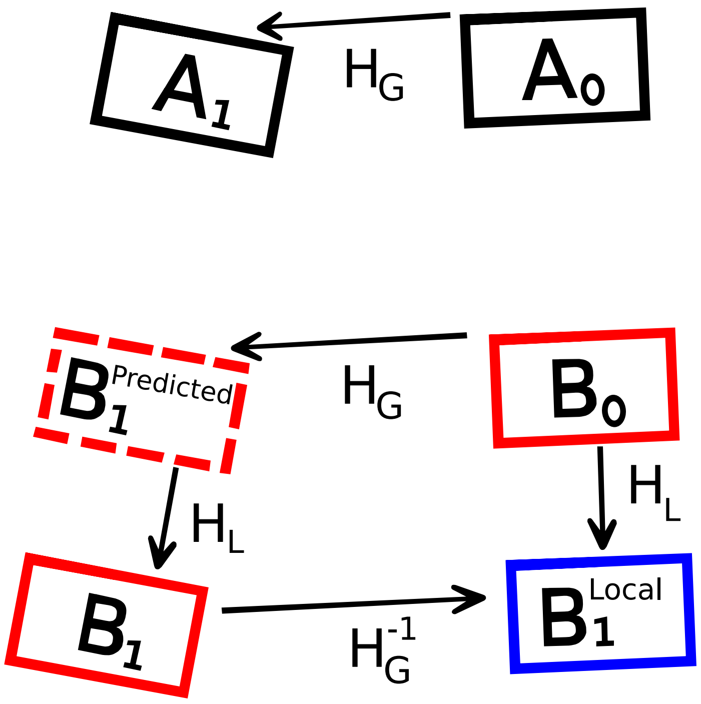


Figure 2: Revealing the local motion between two bodies A and B

If one conducts this process for all observed time frames, a series of local bodies can be obtained. At the end, there is a new body, called local body, which consists of a series of trajectories and represents the local motion between body A and B at each time frame. This is exactly, what was shown in the beginning in figure 1 (the red trajectories). From two apparent unrelated motions, the circular motion, which is an inherent property of a revolute joint, can easily spotted. The next step is to characterize a motion by attributes like circular or linear and classify them accordingly into different categories of joints. But before this is done, a brief discussion about the choice of the reference body follows.

Algorithm 3 getLocalBodyMotion

```
Require: $(A)_{i=1}^{M}, (B)_{i=1}^{M}$
Ensure: $(B_{Local})_{i=1}^{M}, H_L$

 1: $H_G(0) \leftarrow eye(4)$
 2: $H_L(0) \leftarrow eye(4)$
 3: $B_{Local}(0) \leftarrow B_2(0)$
 4: $H_{complete} \leftarrow eye(4)$
 5: for $i = 0$ to $M - 1$ do
 6:     $H_G(i+1) \leftarrow computeHomogenousTransformation(A(i), A(i+1))$
 7:     $H_{complete} \leftarrow H_G(i+1) \cdot H_{complete}$
 8:     $H_{G,inv} \leftarrow inv(H_{complete})$
 9:     $B_{Local}(i+1) \leftarrow applyTransformation(H_{G,inv}, B(i+1))$
10:     $H_L(i+1) \leftarrow computeHomogenousTransformation(B_{Local}(i), B_{Local}(i+1))$
11: end for
```

5 Analysing Relationships

5.1 From local motion to joint types

Once the problem of revealing the relative motion is solved, the right type of joint has to be determined. In the current approach, a distinction is made between four possible types. One is rigid, that means, both bodies are rigidly connected and are basically one body. Then there are the two one degree of freedom joints, prismatic and revolute. They are represented by a translational and a rotational component, respectively. If no one of the mentioned types is present, the connection is declared as disconnected. That means, there could be a possible joint with more than one degree of freedom or there is absolutely no connection between them. The following section discusses, how to distingush between those four types of connections. As a solution, expert-designed joints are proposed as an apriori model to match the observations.

A crucial aspect in dealing with real world applications is the amount of noise, which has to be handled. There are several possible sources of noise depending on the kind of observation process. First of all, image features in the scene have to be found. Because input devices are far from being perfect and are contrained by the physical nature of waves, approximations have to be made. Another source of higher uncertainty is given by environmental changes like illumination, partly hidden objects or fast movements. As in detail discussed in section 3.3, acquiring the input from one camera only, also gives rise to scale ambiguity and also to uncertainty in determing the right depth value. Finally, almost every camera has a white ground noise, which comes for example from fluctuations in the number of electrons in a Charge Coupled Device (CCD) Camera. To handle higher amounts of noise, confidence intervals around the fitting models are introduced.
Altogether two models are used for determing prismatic joints and revolute ones. The first is a cylinder, which will represent the motion of a prismatic joint. The mathematical algorithms and their application will be discussed in section 5.2. The second model is a circle, which represents a revolute joint. How this works is part of section 5.3. At the end, two confidence values are obtained from the models. Each one of them determines, how sure the algorithm is about a specific type of joint. Together with both values, a third number will be acquired, which determines the score about a rigid connection between two bodies. The last part in section 5.4 deals with the determination of the final joint type, given all three confidence values.

5.2 Cylinder fitting

The problem which will tried to be solved in this section is how one can distingush a relative prismatic motion from any other motion. As it is known, a prismatic joint is defined by a translation component only (see figure 3). Common examples from our environments involve drawers or sliding elevator doors. All points on an object move into the same direction and with the same length if they represent a prismatic motion. One intuitive approach would be to measure the length and the direction of each point trajectory. If the all are nearly the same, it is prismatic, otherwise it is

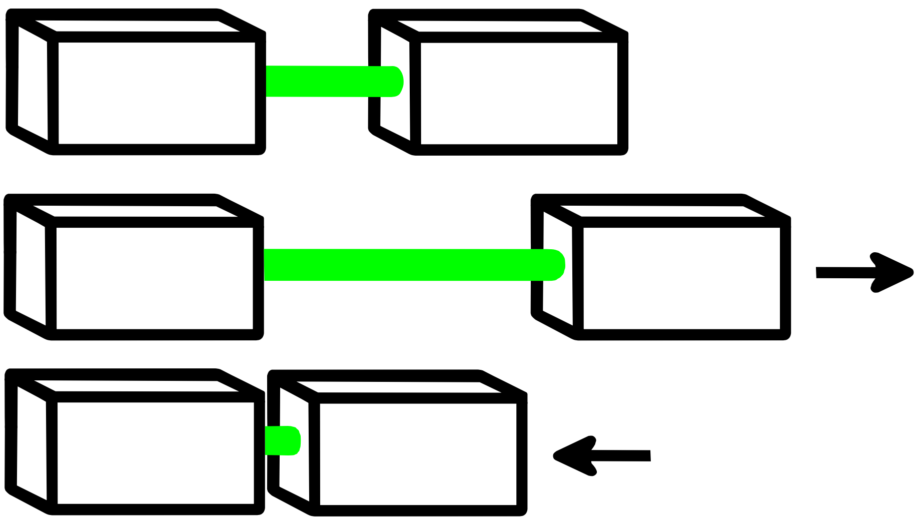
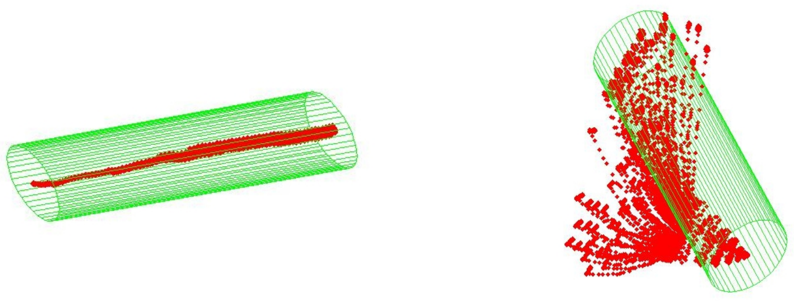


Figure 3: Two bodies connected with a prismatic joint. A transformation between them can be described by a translational component at the axis (green) only.

not. Measuring the direction of a trajectory is easy, if the motion is perfect. But once there is noise, the direction can differ a lot. Random Sample Consensus (RANSAC) algorithms [5] are usually used to infer the right model of the points. But because they are computational expensive, a simpler approach is proposed. Imagine, that all the trajectories would have the same starting point. Even if there is noise, all the points should almost stay on a straight line, which represents the translation component. To deal with small outliers, a confidence interval is introduced around this straight line to form a cylinder. If all the points, representing the trajectories, are inside this cylinder and the length is equal, it is likely, that a prismatic motion is present.
Next all these ideas are summarized in mathematical notation and fit into an algorithmic form. In general, each trajectory $T_{1,..,M}$ of a body is represented by a number of points $P_{1,..,N}$, where M denotes the features on an object and N is the number of timeframes. A feature is an interesting region on an object, which is tracked over time. The summation about all timesteps forms the trajectory. By aligning all the trajectories to have the same starting point, each trajectory is shifted in a way to be aligned with a reference feature. Note that this can be an arbitrary feature, because for a perfect prismatic motion, all trajectories have the same length and same direction. In the developed algorithm, the feature with the longest trajectory was choosen.
Suppose, all the trajectories should be aligned with respect to trajectory T_R, where R stands for reference. The shifted trajectories can be calculated by $T_{1,..,M}^{Shifted} = T_{1,..,M} - T_R$. That means for each point on a shifted trajectory $T_i^{Shifted}(P_j)$ it is calculated

$$T_i^{Shifted}(P_j) = T_i(P_j) + (T_i(P_j) - T_R(P_0)) \qquad (13)$$

For clarification see figure 5. Once all the trajectories are shifted, the next step is to compute a cylinder around them. Therefore, the reference point P_{start} is taken as the



(a) (b)

Figure 4: Cylinder fit to two different motions, a prismatic one (a) and a disconnected one (b)

start value of the line. The end point P_{end} is defined by the point, which has the greatest distance to the starting point, on the same trajectory. A line between those points is naturally defined in parametric form by $p = \vec{P}_{start} + \lambda(P_{end} - \vec{P}_{start})$. Let the length of the line be the euclidean distance between the end points $l = \|\|\|P_{end} - P_{start}\|\|\|$. A confidence interval is obtained, by calculating a tube with a radius equal to a predefined percentage of the overall motion. This percentage relies on the real world performance of the algorithm. In several experiments a value of $p = 0.1$ was used as a robust design parameter for prismatic models. A typical outcome can be seen in figure 4, where a cylinder is fitted to a shifted disconnected motion. The probability of a prismatic joint is now defined by points in the interval divided by all points avaiable. The location of a point inside the interval is obtained by calculating the distance from the line. As described in [25], in 3d space this can be done by calculating

$$d_i = \frac{\|(P_i - P_{start}) \times (P_i - P_{end})\|}{\|P_{end} - P_{start}\|} \tag{14}$$

where d_i denotes the distance of point P_i from the line and $\times$ is the cross product between two vectors. If this value is smaller than the radius of the interval, $r = l * p = \|\|\|P_{end} - P_{start}\|\|\| * 0.1$, the point is counted as being inside the tube. The probability of points in the cylinder is therefore given by

$$p_{cylinderfit} = \frac{K}{N} \tag{15}$$

where K denotes the number of points in the cylinder and N represents the total number of points.
But there is still one problem with this approach. If the cylinder is very long and a disconnected motion is only visible at the end, the confidence level of a prismatic joint would still be very huge. Therefore, the cylinder is divided into smaller parts. In each of those parts, the number of points is counted and divided by the total number of points in this section. If one of the sections has a very small score, the overall confidence also decreases. To achieve this goal, the cylinder is divided into ten equal parts and

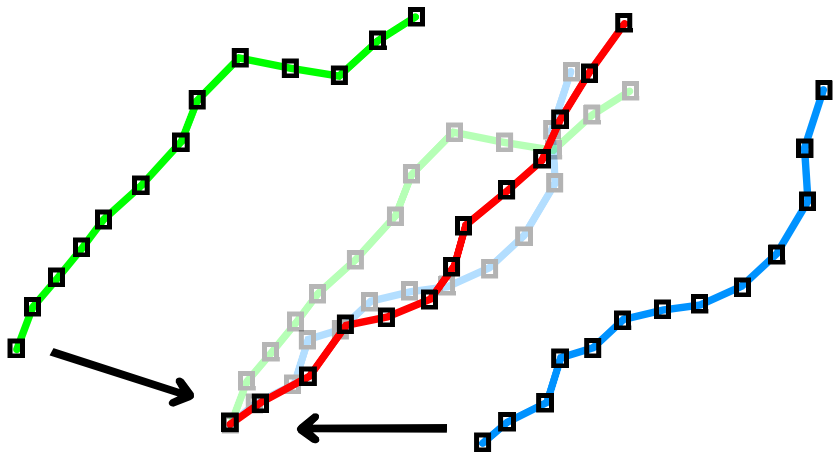


Figure 5: Two trajectories of two different features (blue, green) are shifted, to be aligned with a reference feature (red).

each score is calculated. The overall score is eventually calculated from the particular scores by choosing the minimum value. That assures that the motion will be detected as disconnected, if just the end gives a hint about the true motion[2].

As already discussed, another indice for a prismatic motion is the equalness of trajectory lengths. Therefore, each trajectory is measured. This is done, by calculating two points on the trajectory which have the greatest distance between them. This distance is saved in a list $L_{1,\dots,M}$. For a total of M features there are M entries in the list. To compare these entries to other motions, the list is normed, by dividing each entry by the highest value. The outcome is therefore a list, where one of the entries is one (the maximum trajectory length) and all the other entries represent the percentage of the maximum length. That means, if all the lines are equal, the list consists of ones only. To distingush between equal lines and unequal lines, a sigmoid function is used. Therefore, first the mean value of the line length is calculated. This is done by calculating $\mu_L = \frac{\sum_{i=1}^{M} L_i}{M}$. From this value a confidence score is computed about of how equal the lines are. A high mean value around one gets a high value of near one, while a low mean value gets a lower value. For this need, the Q-function [26] is used which is defined as

$$Q(x) = \frac{1}{2}\left(1 - \operatorname{erf}\left(\frac{x}{\sqrt{2}}\right)\right). \tag{16}$$

[2]It seems to be necessary to extend this part, because the number of ten parts cannot be sufficient under different circumstances. While this has not occurred in experiments, an adaptable number of parts would lead to a higher degree of robustness

where erf is the error function [2], which is defined as

$$erf(x) = \frac{1}{\sqrt{2\pi}} \int_0^x e^{-t^2} dt \tag{17}$$

To represent the above mentioned specification, the Q function is shifted to a value μ and scaled with a variable λ. The resulting equation represents the confidence of equalness with two parameters.

$$Q(x, \mu, \lambda) = \frac{1}{2}\Big(1 - \mathrm{erf}\Big(\frac{(x-\mu)\cdot\lambda}{\sqrt{2}}\Big)\Big). \tag{18}$$

Through experimental testing, values of $\mu = 0.93$ and $\lambda = 40.0$ were chosen. For a set of lines L with mean value μ_L, the confidence that all the lines are equal is given by

$$p_{linesequal} = 1.0 - Q(\mu_L, \mu = 0.93, \lambda = 40.0) \tag{19}$$

To obtain a total score, of how strong the confidence of a prismatic joint is, both indications are combined. The total confidence score is computed by

$$p_{prismaticmotion} = p_{linesequal} \cdot p_{cylinderfit} \tag{20}$$

A high score is obtained, if both the values of equal lengths of the trajectories and the one for the cylinder fit are high. This confidence relies on two parameters, one the radius of the tube, the second the mean value for the Q function, which denotes together with the λ value, if all the lines are equal.

5.3 Modeling revolute joints

The essential principle of distingushing revolute motion and any other motion consists of basically three parts. In the first one, each trajectory is projected onto a unit circle. If the trajectory exerts a rotational motion only, this should also be visible in the projected data. To quantify the goodness of the fit, the second part is dedicated to describe the circle fit in terms of confidence values. The last part tries to find more indications that there is really a revolute joint present. This is done by determing the similiarity of the center points of each motion and how much motion is observed, compared to the unit circle. In the following paragraphs, this fitting process is described in detail. The core concept is to obtain a fit for one trajectory and repeat the process for all existing trajectories afterwards.
It begins by first fitting a plane through each observed trajectory. This is achieved, by calculating the parametric equation of the plane:

$$Ax + By + Cz + D = 0 \tag{21}$$

For each point on the trajectory, following equation is evaluated

$$A_i x + B_i y + C_i z + D = R_i \tag{22}$$

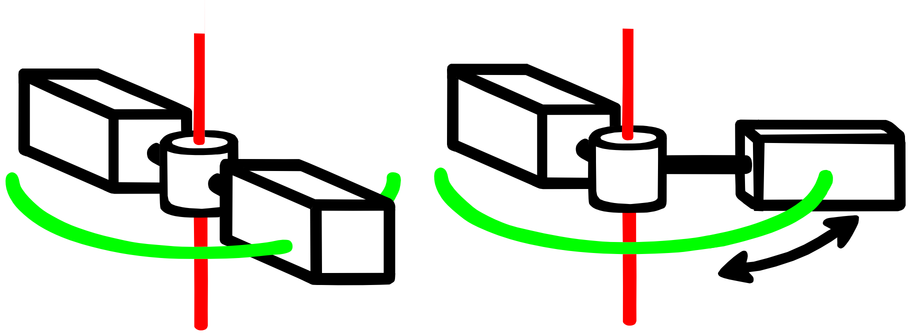


Figure 6: A revolute joint between two bodies. The motion of one of them can be described in terms of a rotational component only (green). The rotation is defined with respect to an axis, where the distance of all points on the rotated body remains constant (red).

To obtain (21), the goal is to minimize the sum about all squared R_i's. From the input data the parameters A, B, C and D have to be determined. Therefore, at least four input points are necessary in order to specify a solution. If the C value is also eliminated by dividing the whole equation by it, the number of points can be decreased to three. This is why at least three points, or time frames, have to be provided, to obtain a solution. Because input data is prone to noise, more input frames will help to obtain better results. To minimize the distance of each point to the plane, a least square distances error function is defined in following way

$$f(A,B,C,D) = \sum_{i=0}^{N} \frac{\|Ax_i + By_i + Cz_i + D\|^2}{A^2 + B^2 + C^2} \tag{23}$$

The following descriptions are loosely based on the notes in [12]. One way of obtaining a minimum value is by minimizing the function with respect to D. Therefore, the first derivative is set to zero

$$\frac{df(A,B,C,D)}{dD} = 0 \tag{24}$$

From this equation, D can be obtained as

$$\begin{aligned} D &= -(A\frac{\sum_{i=0}^{N} x_i}{N} + B\frac{\sum_{i=0}^{N} y_i}{N} + C\frac{\sum_{i=0}^{N} z_i}{N}) && (25)\\ &= -(Ax_0 + By_0 + Cz_0) && (26)\end{aligned}$$

where x_0 represents the mean value of the x-coordinate of the data set and y_0,z_0 the corresponding ones for the y-coordinate and z-coordinate. This leads to a new f function, which contains three parameters

$$f(A,B,C) = \sum_{i=0}^{N} \frac{\|A(x_i - x_0) + B(y_i - y_0) + C(z_i - z_0)\|^2}{A^2 + B^2 + C^2} \tag{27}$$

Equation 27 can be rewritten in matrix form as

$$\begin{aligned} f(v) &= \frac{(v^T M^T)(Mv)}{v^T v} && (28) \\ v &= \begin{pmatrix} A \\ B \\ C \end{pmatrix} && (29) \\ M &= \begin{pmatrix} x_1 - x_0 & y_1 - y_0 & z_1 - z_0 \\ \vdots & \vdots & \vdots \\ x_N - x_0 & y_N - y_0 & z_N - z_0 \end{pmatrix} && (30) \end{aligned}$$

where $f(v)$ can be rewritten as

$$\begin{aligned} f(v) &= \frac{(v^T M^T)(Mv)}{v^T v} && (31) \\ &= \frac{v^T (M^T M) v}{v^T v} && (32) \\ &= \frac{v^T K v}{v^T v} && (33) \end{aligned}$$

where K is equal to $M^T M$. From [15] it is known, that the expression $f(v) = \frac{v^T K v}{v^T v}$ is also called Rayleigh quotient, which is equal to λ_i, if the equation $Kv = \lambda_i v$ holds true. That means, that $f(v)$ approaches the eigenvalue of the matrix K. If K is divided by N, the number of input points, it is equivalent to the covariance matrix of the point set. Once this is known, the normal vector of the plane, which minimizes the function, can be obtained from the eigenvector, which corresponds to the smallest eigenvalue of K. To sum up, the eigenvector v contains the searched parameters A, B, C. From (26), eventually D can be computed. The resulting plane consists therefore of

$$\begin{aligned} A &= v_{\lambda_{min}}(0) && (34) \\ B &= v_{\lambda_{min}}(1) && (35) \\ C &= v_{\lambda_{min}}(2) && (36) \\ D &= -(Ax_0 + By_0 + Cz_0) && (37) \end{aligned}$$

where $v_{\lambda_{min}}$ refers to the eigenvector with the smallest eigenvalue. Once the plane parameters are calculated, the plane and all the features can be projected onto the x-y plane. Therefore, the task is to find a transformation matrix which accomplishes this projection. This can be done in different ways.

The normal vector of the plane is already given by its parameters A, B and C. What basically has to be done is to align this vector to the z-axis. Then the plane has to be translated, so that the origin of the vector coexists with the origin of the coordinate frame. In the implementation, this is done in the following way. First, the angle α between the z-axis and the vector prjected into the z-y plane is calculated. Second, between the z-axis and the projected vector in the z-x plane an angle β is calculated.

Finally, the rotation matrix is build from a rotation about the z-axis with angle α and another one about the x-axis with angle β. This matrix, which rotates the normal to be aligned with the z-axis, is now calculated by multiplying the rotation around z onto the rotation around x. This can be visualized as first rotating the normal onto the z-y plane by rotating it about the z-axis. Once it is located in the z-y plane, a rotation about the x-axis will be sufficient to align the vector with the z-axis. After this is accomplished, all features are multiplied by this rotation matrix. The result is a planar distribution of features parallel to the x-y-plane with a specific offset. This offset is nearly similiar for all features. Therefore, the z-value of the first features is used, to translate the set of features onto the x-y-plane.
With all the features lying on a planar surface, a two dimensional circle fitting process can eventually be realized. Using a least square circle fit algorithm [23], the radius r_c and center points c_x and c_y can be estimated. Given those three parameters, the trajectory of each feature can be projected onto the unit circle in the x-y plane. The new data points are given by

$$x_i^p = \frac{x_i - c_x}{r_c} \tag{38}$$

$$y_i^p = \frac{y_i - c_y}{r_c} \tag{39}$$

To estimate a score, of how good the data is fitted by this unit circle, the distance of each data point to the origin is calculated. This is done by the euclidean distance

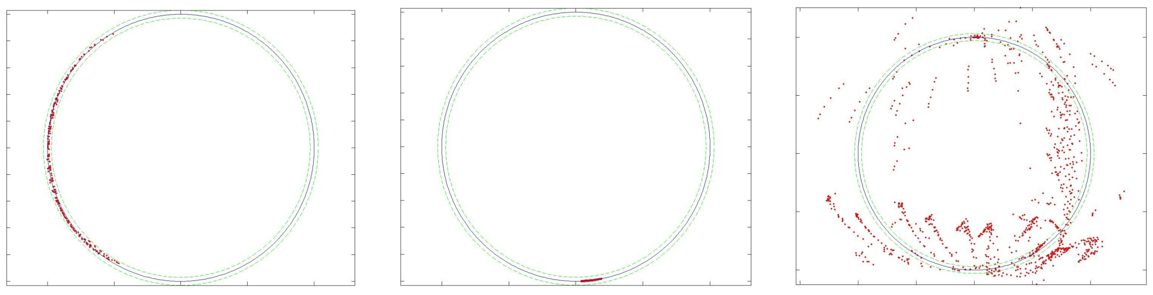
$$d_i = \sqrt{(y_i^p)^2 + (x_i^p)^2} \tag{40}$$

For a perfect circle fit, all the d_i values should be one. But because of noise, this is not true, even for a perfect revolute motion. Therefore, a confidence interval with a parameter ci is introduced. If the distance of the point d_i is located in the interval $[1.0 - ci, 1.0 + ci]$, it is declared as a good circle fit. In experiments, this value was chosen to be $ci = 0.03$. A score for each trajectory is obtained by counting the number of points f_{in}, which are located inside this interval. This value is divided by all points on the trajectory f_{all} to obtain the score

$$p_{circlefit} = \frac{f_{in}}{f_{all}} \tag{41}$$

Examples of fits to a revolute motion can be seen in figure 7(a) and as a comparison for a disconnected motion in figure 7(c). Note that the difference between disconnected and revolute can be smaller as in the examples, especially in the case, where only a small amount of data is collected. But this is exactly what is desired: if not enough evidence is avaiable, it is a better choice to declare a disconnected motion as revolute, instead the other way around.

This approach has a weakness. Consider for example a straight line segment. With the current fitting process, this can be modeled by a very huge circle. As an example, consider figure 7(b), where a huge circle fits a straight line segment very well. Therefore,



(a) (b) (c)

Figure 7: Circle fitting to three different motions. All the trajectories are projected onto the unit circle. Each red dot represents a feature at a specific instance in time. The images are caused by (a) a revolute motion, (b) a prismatic motion, where a huge circle is fitted to a straight line segment and (c) a disconnected motion.

the knowledge about the length of the motion with respect to the circle, provides essential informations about it. As a measurement of length with respect to the circle, the angle of the motion on the unit circle is calculated. This is done in the following way. First, the angle between each point of the trajectory and the x-axis is calculated. This angle is discretized to represent an integer in a 360 degree list. If for example a value of 271.38 degree is obtained, the entry of 271 is increased. If enough points are given, the list should consist of two areas. One with zeros only, because no point is located at this angle position. The other one, the area, where the points are centered. The occupied angle can afterwards be calculated by computing the greatest area, where only zeros are present. This value is subtracted from 360 to form the occupied angle. Each trajectory will therefore have a occupied angle α_i. Again, the Q function (see (18)) is used to assign a score.

$$p_{occupiedangle} = 1.0 - Q(mean(\alpha_{i=1}^{M}), \mu = 20.0, \lambda = 0.5) \tag{42}$$

Where *mean* calculates the arithmetic mean value of all the observed α_i values for M trajectories. The values $\mu = 20.0$ and $\lambda = 0.5$ are chosen as part of the expert designed joint types.

So far, only a confidence value, representing the goodness of fit was calculated. But one also wants to know the axis of rotation for the two bodies. As a byproduct of the circle fit, the center points c_{x_i} and c_{y_i} are known for each trajectory. If a revolute joint is present, there is a line, which can fit the center points.

This line is obtained by computing the Principle Component Analysis (PCA) for the shifted center points. A good introduction into PCA is provided by [19]. Here, the PCA will be used, to rotate the original coordinate frame in such way, that the distance of all points with respect to the x-axis is minimized. To shift the points, the geometric mass of the set is computed. This is given by the mean value of the data set. Given this value, all points are shifted in a way that the mean equals the origin. After computing the PCA, the obtained x-axis points into the same direction as the axis of rotation[3]. This is used to define the well known parametric form of a line $l = \vec{p} + \lambda\vec{d}$, where $\vec{p}$ is the position vector and $\vec{d}$ the direction vector. $\vec{d}$ is given by the x-axis of the PCA and $\vec{p}$ by the translation vector between the mean of the data points and the origin. While it is possible to use the variance of the points with respect to the x-axis, in this case, a cylinder around the line is used to define the goodness of the line fit. As in the case of the prismatic motion, a cylinder is fitted around the line. The number of points in this cylinder f_{incyl} divided by all points f_{all} leads to the score for the line fit of the axis of rotation.

$$p_{axisfit} = \frac{f_{incyl}}{f_{all}} \tag{43}$$

Once this is obtained, the axis of rotation is known and three scores are calculated to define the confidence score that the revolute models fits the data. The overall confidence can afterwards be calculated by

$$P_{revolutemotion} = P_{circfit} \cdot P_{occupiedangle} \cdot P_{axisfit} \tag{44}$$

One final remark considering the revolute fit is about the computation of the PCA. This algorithm always returns a solution, independent of the distribution of the center points. One problem, which can arise in the case of a planar object is a point distribution of the center points. Consider a turntable, where all the center points are concentrated in the middle of the plate. In this case, there are an infinite number of lines, which would make perfect sense from the PCA point of view. Therefore, if there is one plane, which can fit all trajectories, the PCA is not the best choice. The normal to the plane, located at the center point would be the best approximation of the axis of rotation. Nevertheless, in the current implementation this is still neglected, but is obvious an important feature for future improvements.

5.4 Confidence scores

The confidence scores for each joint are computed by taking into account the specific values from the two model fits. Additionally, another score, representing a rigid motion, is calculated. Given the motions of the initial bodies and their relative motion, the travelled distance for each one of them is calculated. This one is simply given by the sum about the euclidean distances between all time frames. If there are two bodies A and B, the three scores are given by $dist_A, dist_B, dist_R$, where R denotes the distance of the relative motion. In order to calculate the confidence score, first following fraction is calculated:

$$f = \frac{dist_R}{\min(dist_A, dist_B)} \tag{45}$$

It states if $f \gg 1$, the two bodies are not rigidly connected. On the other side, if the fraction is small $f \ll 1$, it is likely, that both bodies belong together, or not enough evidence is present to declare another type of joint. To describe this behaviour,

[3]This is only true, if the points a really representing a line. This issue will be discussed at the end of this section.

a modified gaussian is used. It should contain the properties, that for $f \simeq 0$, the confidence value is one. For $f \gg 1$, the confidence value converges to zero. With a standard deviation of $\sigma = \frac{1}{\sqrt{2\pi}}$, a mean value of $\mu = 0$ and an additional parameter λ, which defines the shape, following function can be obtained.

$$p_{rigidmotion} = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{-\lambda \cdot f^2}{2 \cdot \sigma^2}\right) \tag{46}$$

$$= \exp\left(-\lambda \cdot f^2 \pi\right) \tag{47}$$

where f is the fraction from (45). In figure 8 this function is plotted for fractions in the interval $[0, 1]$ and different values of λ. The choice of λ in (47) depends on how conservative one will be about motions. If λ gets bigger than one, only motions, where almost no relative motion was observed are declared as rigid. On the other side, for smaller λ values, even big relative motions are declared as rigid. To achieve a reasonable tradeoff, λ is choosen to be 3. This is the representation of the green line in figure 8.

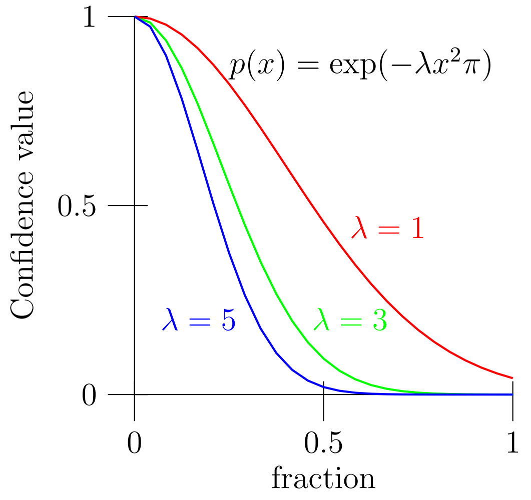


Figure 8: Function used to represent the confidence state of a rigid motion. If the fraction is small, it is likely that the bodies are rigidly connected. When a motion is observed, the fraction grows larger and the confidence will eventually converge to zero. The influence of the parameter λ can be seen for $\lambda = 1$ (red), $\lambda = 3$ (green) and $\lambda = 5$ (blue).

Once the last value of the confidence state of a rigid motion is obtained, the overall confidence scores can be computed. They represent the final state of the algorithm. First, the confidence score of a rigid connection can directly be obtained from the one of the rigid motion. The model fit values are not important any more, because no motion is present. This is why, the rigid confidence score can be written as

$$p_{rigid} = p_{rigidmotion} \tag{48}$$

Second, the prismatic confidence score is computed as a simple if-like statement. If no rigid motion is present and the prismatic motion score is high, also the prismatic score should be high. From this consideration it follows, that one can compute the confidence score as

$$p_{prismatic} = (1 - p_{rigid})p_{prismaticmotion} \tag{49}$$

The next joint type parameter also depends on the other two previous calculated values. A revolute joint is likely present, if the rigid and the prismatic scores are low.

$$p_{revolute} = (1 - p_{rigid})(1 - p_{prismaticmotion})p_{revolutemotion} \tag{50}$$

If none of the confidence values got a high score, the joint type is either a higher dimensional one or completely disconnected (six degrees of freedom to describe the motion). At the moment, no other joint type can be obtained. This is why, every other motion is declared as disconnected. This can be written as a score in following manner:

$$p_{disconnected} = (1 - p_{rigid})(1 - p_{prismaticmotion})(1 - p_{revolutemotion}) \tag{51}$$

Those four parameters represent the final confidence state of the algorithm. The detected joint type is choosen as the one, which contains the highest score.

5.5 The scaling problem

A problem which arises, if all the features are obtained by a monocular camera is the scaling problem. Without depth information, the same image can for example be caused by a huge object far away or by a very small object, near to the camera plane. Consider the moving of one object from a time frame t_0 to another one t_1 (figure 9). The distance between the two points A and B in time frame t_0 and between A' and B' in time frame t_1 remains constant. For an observer, tracking those two points, there would be an additional translational component visible. This is why the scaling has to be considered, if one seeks to determine the relationship between those two points. Even if this sounds rather theoretical, it occurs already in the situation, where two drawers are pulled from a drawers cabinet. Take a look at figure 10, which represent the relative motion of one of the drawers at different scales. In red, the trajectory of the relative motion of one single feature at the correct scale (0.96) can be seen. As one would expect, if the scale differs a translational component is added to the relative motion. Note, that a special case occurs when depth information is known. The scale is then already known to be one. That means, no one of two bodies has to be scaled, to find the truth relationship. Any 3d information is already scaled in the right way.
The scaling problem is visible, if a body moves without rotation into a certain direction. If another body moves exactly into the same direction, but not with the same distance, there could be two reasons for this behaviour. One is, that there is a prismatic joint between them which causes the additional translational component. The other reason would be, that this component is caused by scale ambiguity. To understand this, consider again figure 9. The green line describes a transformation T_g and is introduced by scale. Therefore, it can be computed from the transformation T_r by multiplying it with a scaling factor k. The resulting equation gets

$$T_g = k \cdot T_r \tag{52}$$
$$k = T_r^T \cdot T_g \tag{53}$$

Given a longer sequence of different motions, it becomes highly unlikely, that this additional translational component is caused by a prismatic motion. It would mean, that the motion is always observed in the camera plane. But in this case it is better to be conservative. If no direct evidence for a prismatic joint is present, stay with the assumption, that there is no prismatic motion.
Instead of calculating the transformation, one of the two bodies could be scaled. If the scaling factor k is known, multiplying each feature on a body, would reveal the truth relationship between two bodies. But unfortunately, this scaling factor is not known a priori.
To infer the scale, different possible k's are tried. For each one of them, the score for a rigid, prismatic or revolute connection is tested. In theory, only for the right scale, there should be a high score for the right relationship. If the scaling factor is increased or decreased from there on, the additional translational component grows larger. The correlation between scale and scores is not linear, but almost a monotonic increasing function. Therefore it is possible to search for the right scale by means of an optimization method. While the field of mathematical optimization is huge, especially

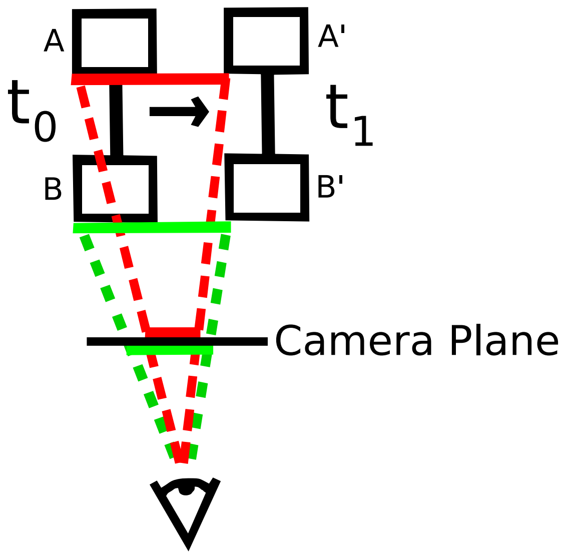


Figure 9: An object is moving from time t_0 to t_1. While the actual distance between two points A and B on the object stays constant, the observed motion differs. The observer could conclude from this observation, that there is a translational component between both of them.

for non linear optimizations, in [17] different basic algorithms can be found, which are a good fit for this problem. In the experiments this whole approach was simplified by testing all scales in the interval $[0.5, 1.5]$ with a step size of $k_s = 0.01$. At each step, the scores for each relationship were calculated. The best scaling factors are usually centered around 1.0. For example, in the laptop experiment, the keyboard has to be scaled by $k = 0.94$ to obtain the revolute motion with respect to its screen. This is shown in figure 11, where the trajectory of one feature at different scales is plotted. As can be seen, only the right scale shows the revolute motion. Another example is the drawers experiment, where a scaling factor of $k = 0.96$ is used for one of the drawers. Figure 10 shows again the motion of one single feature from the relative motion. The results of both experiments can be found in the appendix. Isn practical settings like this, the scale can be constrained in an interval around the special case of $k = 1.0$. This approach lacks the elegance of the optimization method, but could be very useful together with the computational power of mass parallelization on a graphical processor unit (GPU).

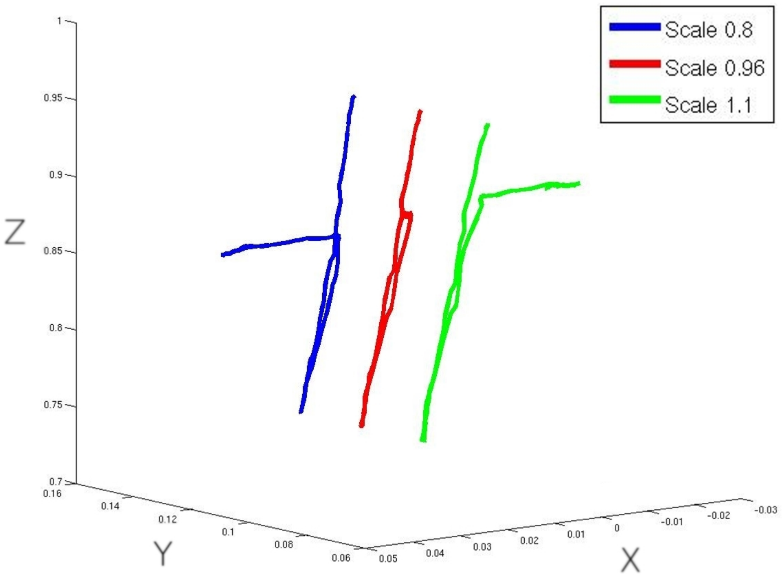


Figure 10: The relative motion between two drawers (just one feature). While the prismatic motion is present as a straight line at the right scale (red), an additional translational component is added for two wrong scale factors (blue and green). Adapted from [10].

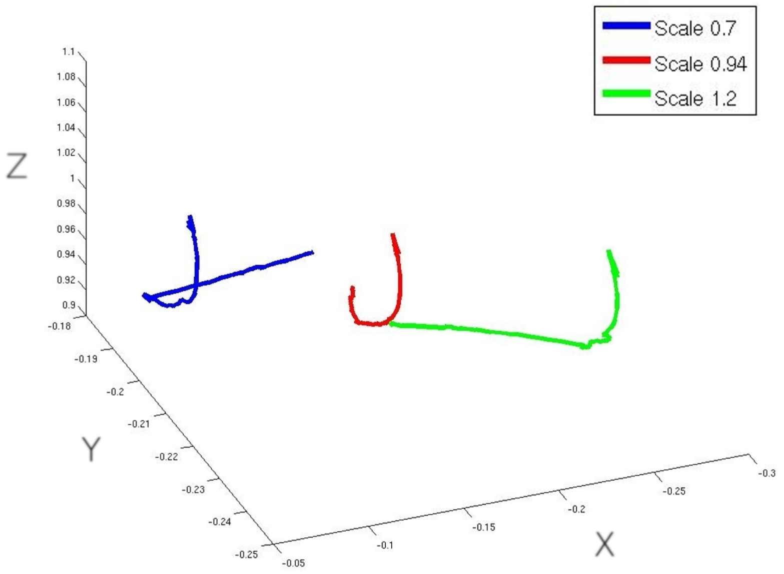


Figure 11: The relative motion between a laptop screen and its keyboard (just one feature). At the true scale of 0.94, the revolute motion can easily be seen (red). At other scales, an additional translational component distorts the result (blue and green). Adapted from [10].

6 Experimental Results

6.1 Real World Experiments

To verify the robustness and correctness of the algorithm one cannot rely on simulated data only. First of all, the degree of noise which is exerted by the input data is not known apriori. Secondly, simulated data always bears the possibility of cheating in a way which is not perceived by the experimenter. Therefore, real world experiments are a great opportunity to test the strength and flexibility of the algorithm.
Altogether ten experiments were conducted. Each experiment was carried out in the following way. A camera observed the scene, while an object is pushed by a robotic manipulator. From the video, features are tracked over time. Given the motion of the features, a clustering is being produced, depending on different predictors, like the distance or features which moved together (see [10] for further predictors to cluster features). The motion also gives information about the depth value. If the distance between features on one cluster changes, the depth of this cluster has most likely changed. This is one way of providing the input for the developed algorithm, but not exclusively the only one. If the input was generated by a motion capturing device, as discussed in the related work section, the output would become clearer. The confidence values, which are shown for each experiment, would be near one.

Once the clustering and 3d reconstruction is done, the 3d features are used as input for the joint detection algorithm. As it will be seen, this data also shows some surprising side effects, which wouldn't be visible if one had relied on simulated data only. The experiments contained doors, tricycles, drawers, moving tables, a laptop, elevator doors, a fridge and a toy train, as can be seen in figure 12.

In each experiment, the feature trajectories and the number of clusters were already provided. For each pair of clusters in the scene, the joint detection algorithm was carried out as described in the last sections. Figure 13 shows the relative motion between two different clusters of a tricycle (wheel and top part). A background cluster was added to show the relationship of each cluster compared to the static environment (not shown in the figure).

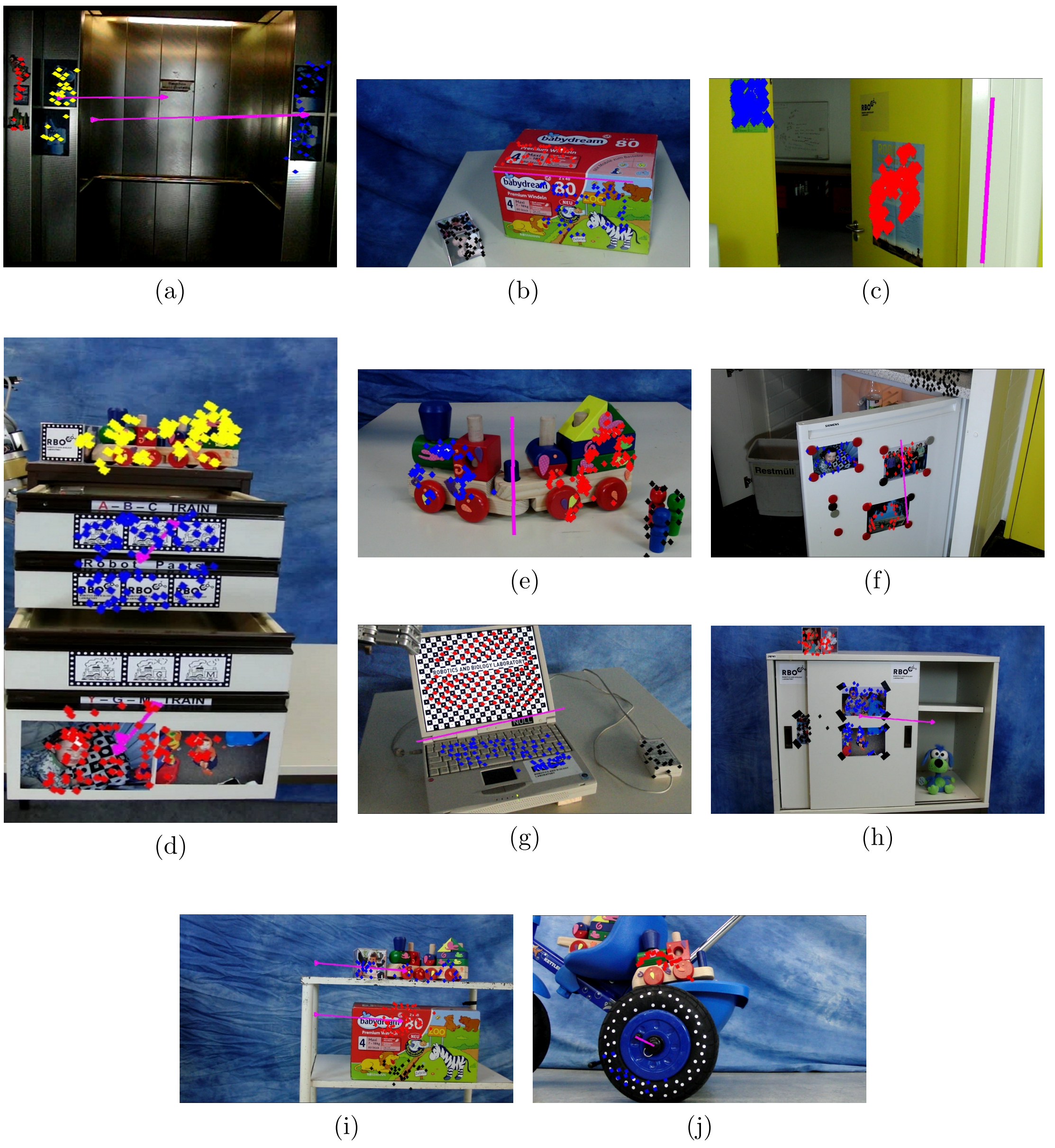
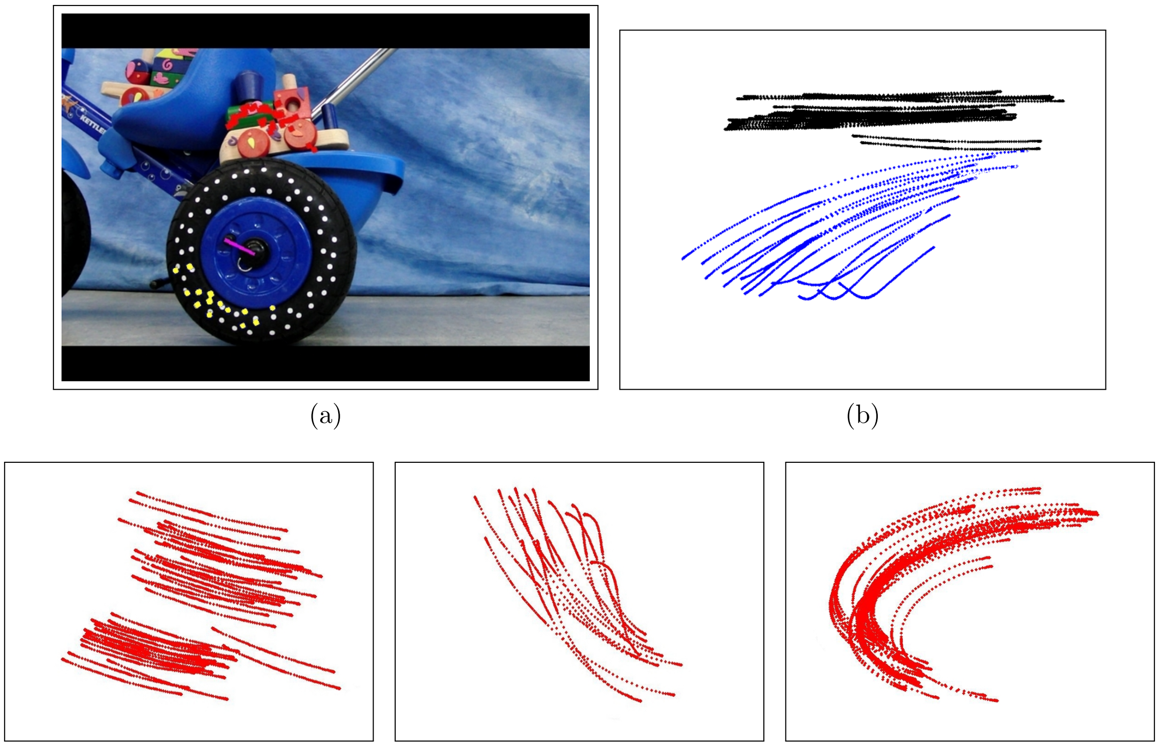


Figure 12: All experimental results: The opening doors of an elevator (a), a diaper box moving on a table (b), an opening door (c), a moving drawers cabinet (d), a fridge door (f), a laptop with moving screen (g), a sliding door of a book shelf (h), a rolling table (i), a pushed tricycle (j) and a pushed train toy (e). Each one of those experiments can be seen in more detail in the appendix section. Adapted from [10].



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

Figure 13: Tricycle Experiment with features on the wheel and top of the bike (a). The tracked 3d data is shown in (b). Given those trajectories, the relative motions are shown in (c) between background and the top part of the tricycle, (d) background and wheel and (e) wheel and top part.

Once the relative motion is revealed, a cylinder and a circle fitting is conducted. Figure 14 shows each relative motion in the train experiment and their cylinder and circle fit results.

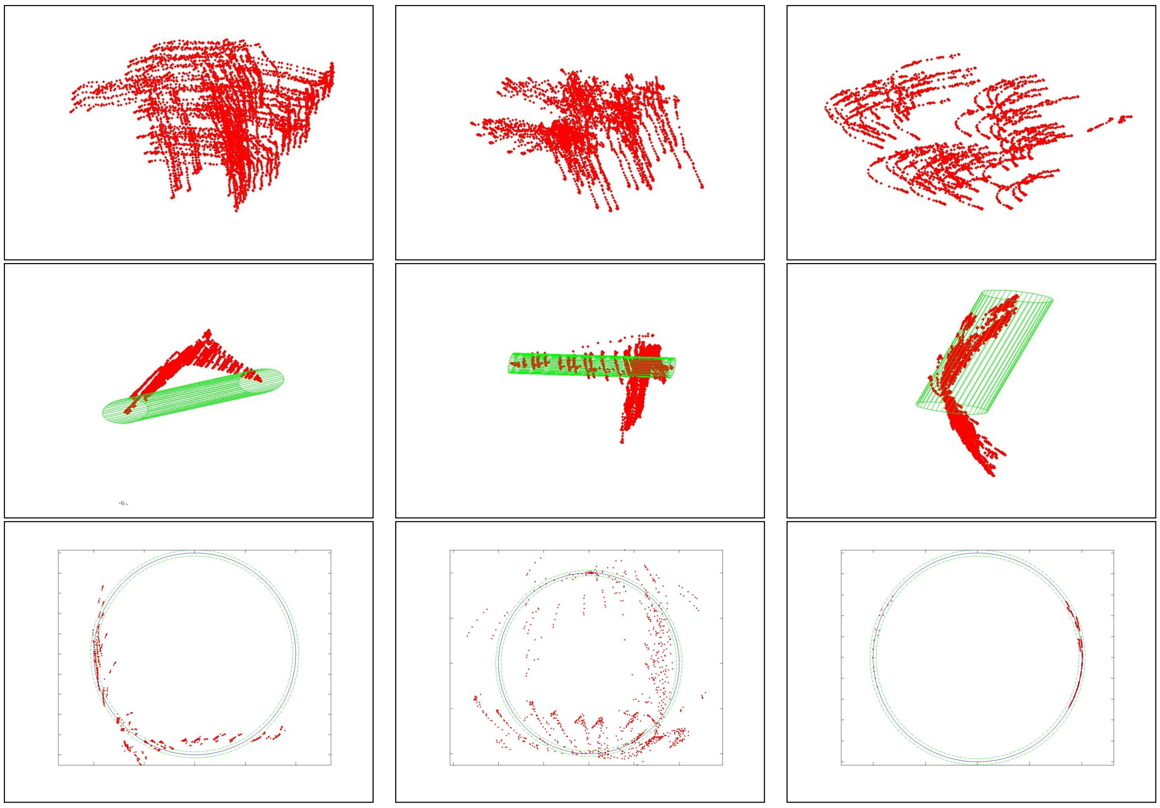


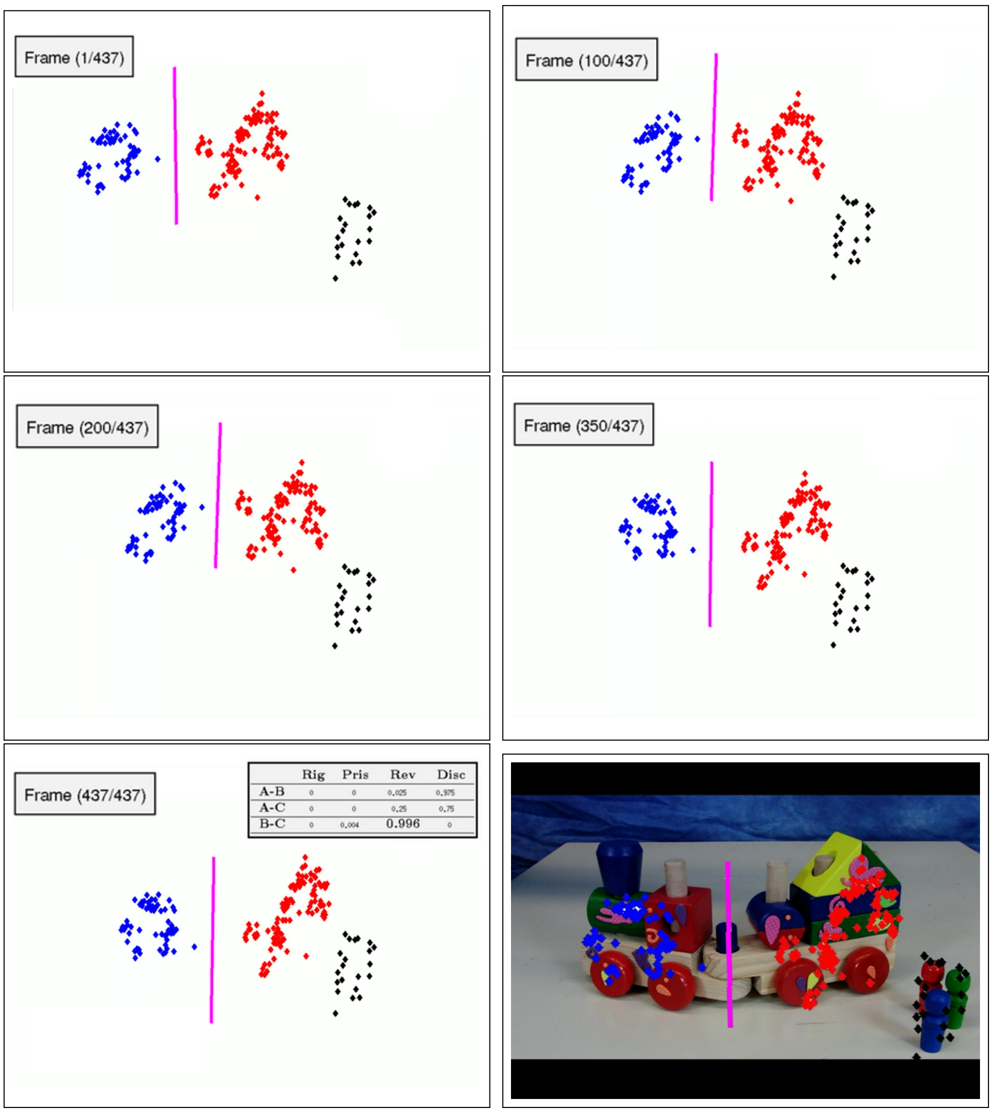
Figure 14: Row (1): Relative motion of the left train compared to the background (left), between background and right train (middle) and between left train and right train (right). Row (2): Cylinder fit to each of the motions. Row (3): Cylinder fit to each motion.

Results for Train Experiment				
Parts	Confidence scores for Relationships			
	Rigid	Prismatic	Revolute	Disc
Background-Left Train	0.000	0.000	0.025	**0.975**
Background-Right Train	0.000	0.000	0.250	**0.750**
Left Train-Right Train	0.000	0.004	**0.996**	0.000

Table 1: Results for Train experiment

The complete result of the algorithm on the train data set is shown in table 1. It can be seen, that both trains are seen as disconnected with respect to the background. Between them, a revolute joint is detected with a confidence score of 0.996. To verify the correctness of this experiment, the resulting axis of the revolute joint is plotted onto the dataset. In figure 15 the axis is shown throughout the motion. Note, that all images in this figure show 3d points over time except the last one. Here the points are projected back onto the camera plane.

Each of the other experiments was conducted in the same manner. A complete overview of them is shown in figure 12, where the resulting axis is plotted in the last frame of the motion. For a more detailed treatise of the experiments, the reader is referred to Appendix A-J. All together 31 relationships were analyzed. The correct type of joint was found in 30 cases. The only error occurred in the fridge experiment, where a revolute joint between the door of a fridge and the environment should be found, but was declared disconnected. This case will be discussed in section 6.2.



	Rig	Pris	Rev	Disc
A-B	0	0	0.025	0.975
A-C	0	0	0.25	0.75
B-C	0	0.004	0.996	0

Figure 15: After calculating the axis of rotation, it is overlayed to each frame of the original input data. For frames (a)1, (b)100, (c) 200 and (d) 350 you can see the results. Frame (e) 437 also shows the score of each relationship in the data.
The process of drawing the axis is automated, because the relative motion algorithm already provides the necessary transformations. (f) shows the last frame of the original video, with the features and the axis overlayed. Note that images (a)-(e) show the points after 3d reconstruction, while (f) is a 2d projection of the points back to the camera plane.

Standard Deviation	Mean Prismatic Probability ($N = 20$)
$\theta = 10$	0.992
$\theta = 25$	0.961
$\theta = 50$	0.769
$\theta = 100$	0.236

Table 2: Performance of prismatic detection with added gaussian noise. At each standard deviation, $N = 20$ experiments were conducted.

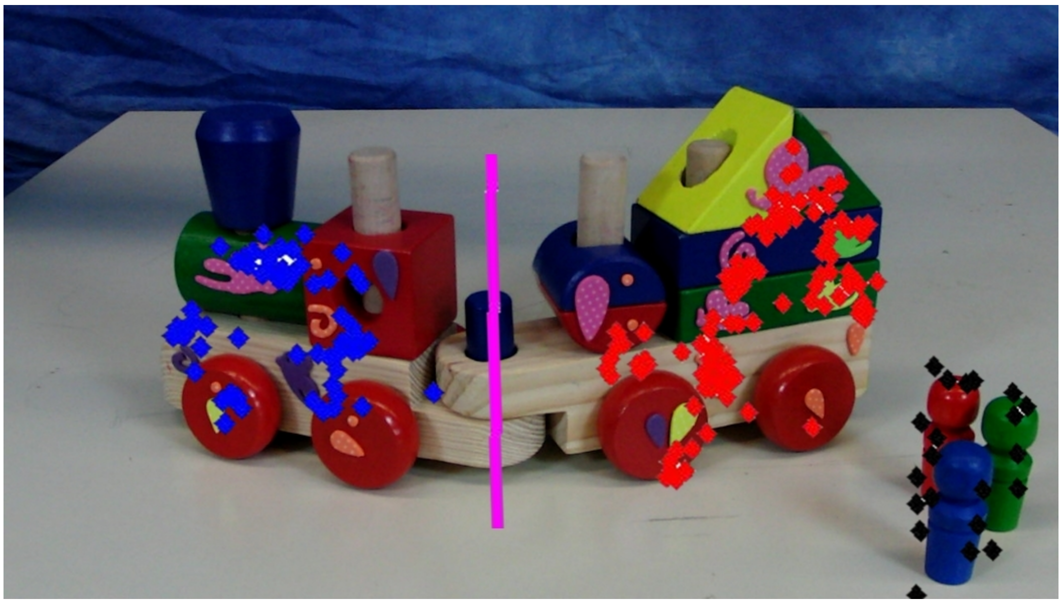
6.2 Limitations

To test the limitations of this approach, a gaussian noise is added to a simulated prismatic motion. This is done, by calculating the overall motion, that is the distance between the first and the last frame of the first feature. Each feature in each time frame gets an additional sample from a gaussian distribution with variance θ and mean zero. To scale this distribution to be consistent with the motion, the sample is divided by the overall motion. To verify the correctness, for a set of variances (10,25,50,100) a constant number of 20 runs is performed. The results are the mean probability of a prismatic motion for each variance. The outcome is listed in table 2. As one would expect, the probability gets worse, if the variance of the gaussian noise is increased.
What does this mean practical for the use of the algorithm? This question can answered best by looking at Appendix 7. This shows the fridge experiment, where its door is opened. Because the input data is very noisy, the right type of joint for one cluster is not correctly detected. The reason can be found in the motion of this cluster. Rather than exerting the circular motion, which is searched for, the motion is elliptical. The algorithm behaves as intended: Because no circular motion is found, the joint is declared as disconnected. If one wants to include joints, which are characterized by elliptical motions, the circle fit can be augmented by an elliptical fit. Again, the paper from Taubin [23] provides the essentials for the computation of this type of fit.
To sum up, the algorithm performs as expected. Model fitting techniques always assert, that one knows what has to be found in the data. Therefore, it is important to specify the constraints of a model in order to achieve the right results.

7 Conclusion

This work presented an approach to identify joint types in 3d point clouds. As discussed in section 3.3, the input cloud is already presegmented and represents an moving articulated object.
To address this problem, the relationship between each pair of clusters in the scene is calculated separately. The proposed algorithm then starts by extracting the relative motion between the different input clusters. Once the individual motion is known, the right joint type is determined by means of different fitting methods. At the end, the position and orientation of a joint is returned together with a confidence score. The knowledge about the joint types can be used further to build a simplified model of a kinematic structure.
As shown in the experimental results, the proposed joint detection algorithm can be used with real world objects. It is robust against noise up to a limit, which was shown in section 6.2. Nevertheless, for 30 out of 31 relationships that were analyzed in the experiments, the algorithm could detect the right type of joint. Still, the proposed approach cannot be seen as a general solution to the problem of analyzing 3d point clouds of rigid objects.
First of all, only prismatic and revolute joints were analyzed. More complex forms of objects cannot be identified with the current approach. Therefore, the work could be extended to handle different types of joints with more than one degree of freedom. This includes for example the analyzis of lower-pair joints like spherical, planar, cylindrical or screw ones. If all lower level joint pairs can reliably be found, almost all articulated objects could be analyzed.
The second shortcome of the algorithm is the growing computational complexity of the solution. Given n clusters in the point cloud, $\binom{n}{2}$ relationships have to be analyzed, which causes an additional complexity of $O(n^2)$. To handle this complexity, a parallelization of the code is another starting point for further improvements.
Finally, a further important topic for future research is the automatic acquisiton of dynamical properties. System identification tasks would greatly benefit from an extracted kinematic and dynamical model of a manipulator.
To sum up, acquiring object structures without previous knowledge is still a hard task to master. As shown, this thesis contributes to the effort of solving this problem by proposing an approach to analyze presegmented 3d point clouds. With respect to the grand scheme, this work is another small building block in order to build robots, which can improve and make our lives easier. The sheer complexity of ordinary tasks still bears ambitious goals for researchers worldwide. If the level of abstractness is continually increased, it could eventually lead to machines, which can do our everyday work.

Appendix A: Train Toy



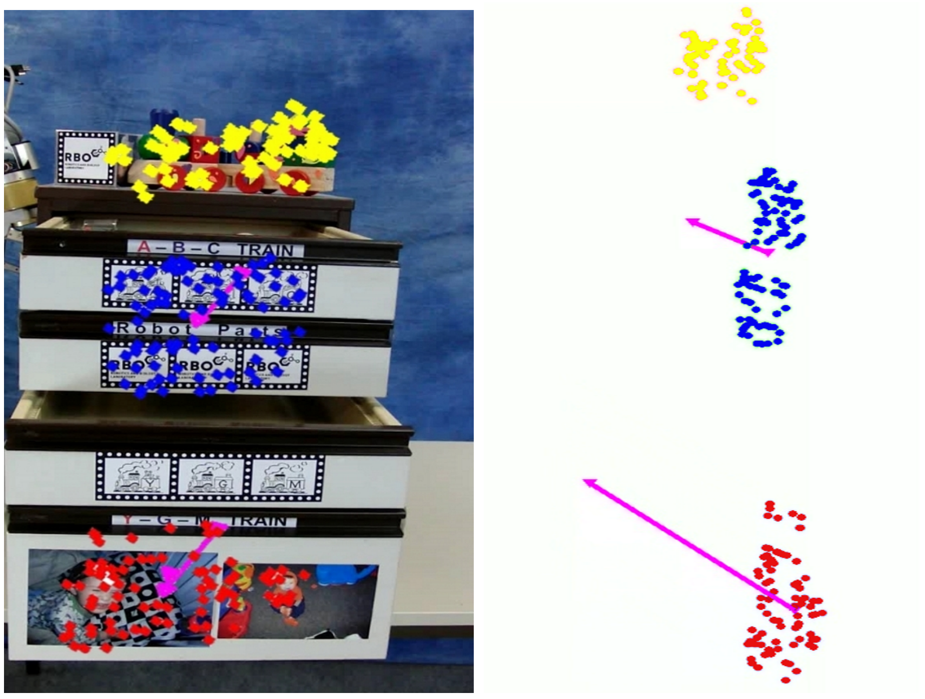
(a) Train Cluster

Results for Train Toy Experiment				
Parts	Confidence scores			
	Rigid	Prismatic	Revolute	Disc
Background-Left Train	0.000	0.000	0.025	**0.975**
Background-Right Train	0.000	0.000	0.250	**0.750**
Left Train-Right Train	0.000	0.004	**0.996**	0.000

(b) Results

Figure 16: Experimental Results for the Train Toy experiment. As discussed in the experimental section 6, a train toy is pushed by the robot arm over the table. The relationship between the left train (blue) and the right one (red) is correctly detected as revolute. The axis of rotation resembles what would be expected from the analysis. Adapted from [10].

Appendix B: Drawers



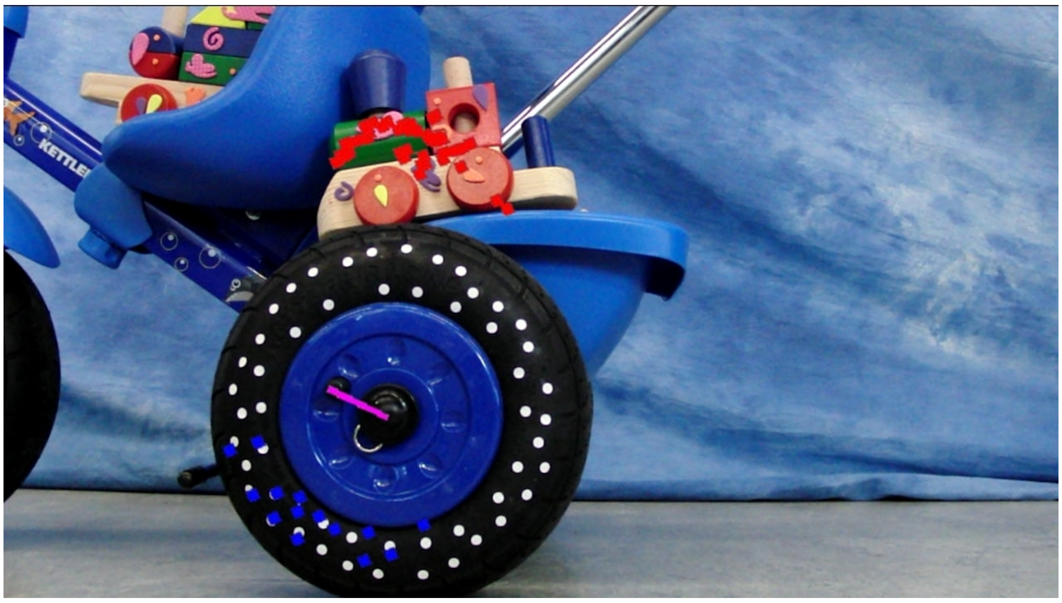
(a) Drawers Cluster

Results for Drawers Experiment				
Parts	Confidence scores			
	Rigid	Prismatic	Revolute	Disc
Frame-Top Drawer	0.000	**0.996**	0.002	0.002
Frame-Bottom Drawer	0.000	**0.975**	0.014	0.011
Top Drawer-Bottom Drawer	0.000	**0.969**	0.022	0.009

(b) Results

Figure 17: Experimental Results for the drawers experiment. Three clusters are used to determine the relationship. The top of the drawers cabinet (yellow), the top drawer (blue) and the bottom drawer (red). Overlayed, the three axes can be seen. Because it is hard to spot them in the image on the left, the features are plotted in 3d space on the right. The point of view is approximately 45 degrees rotated in comparison to the camera plane. The threedimensional structure of the axes can easier be seen. Note, that both axes point slightly downwards. This is exactly what can be seen in the image on the left. From the viewpoint of the camera, it seems, that especially the bottom drawer is pulled downwards. The position and orientation represents what is seen with respect to the camera plane. Adapted from [10].

Appendix C: Tricyle

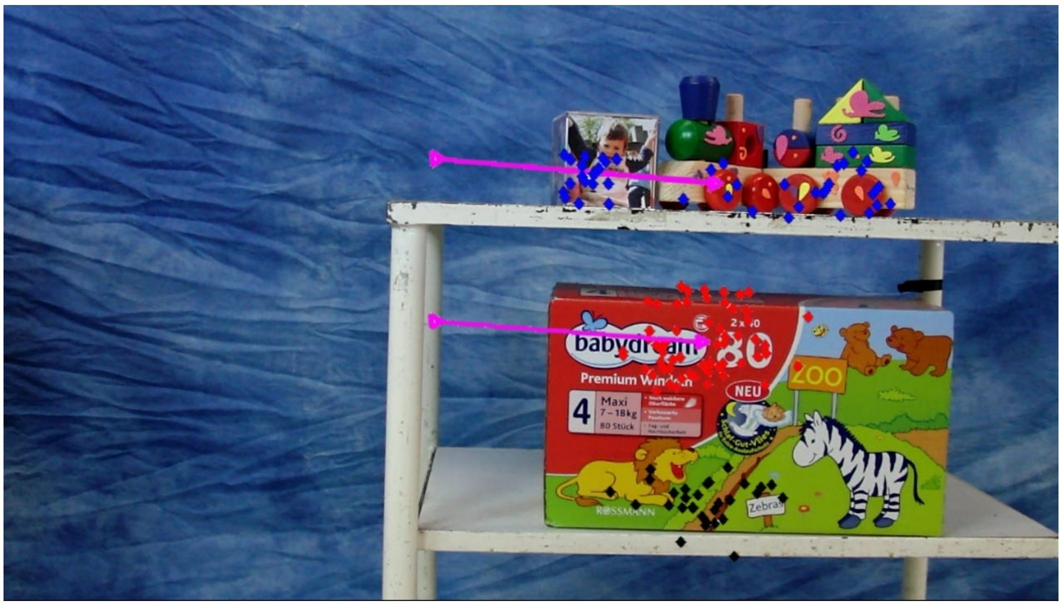

(a) Tricyle Cluster

Results for Tricycle Experiment				
Parts	Confidence scores			
	Rigid	Prismatic	Revolute	Disc
Background-Top	0.000	0.000	0.125	**0.875**
Background-Wheel	0.000	0.000	0.008	**0.992**
Top-Wheel	0.001	0.000	**0.999**	0.000

(b) Results

Figure 18: Experimental Results for the tricyle experiment. A cluster on top of a tricycle (red) and one on the wheel (blue) were analyzed with respect to each other and the background (a non moving static cluster which is not shown here). The table indicates the confidence scores of a specific joint between each pair of clusters. The resulting axis from the revolute case between wheel and top was plotted onto the image (pink). Adapted from [10].

Appendix D: Table



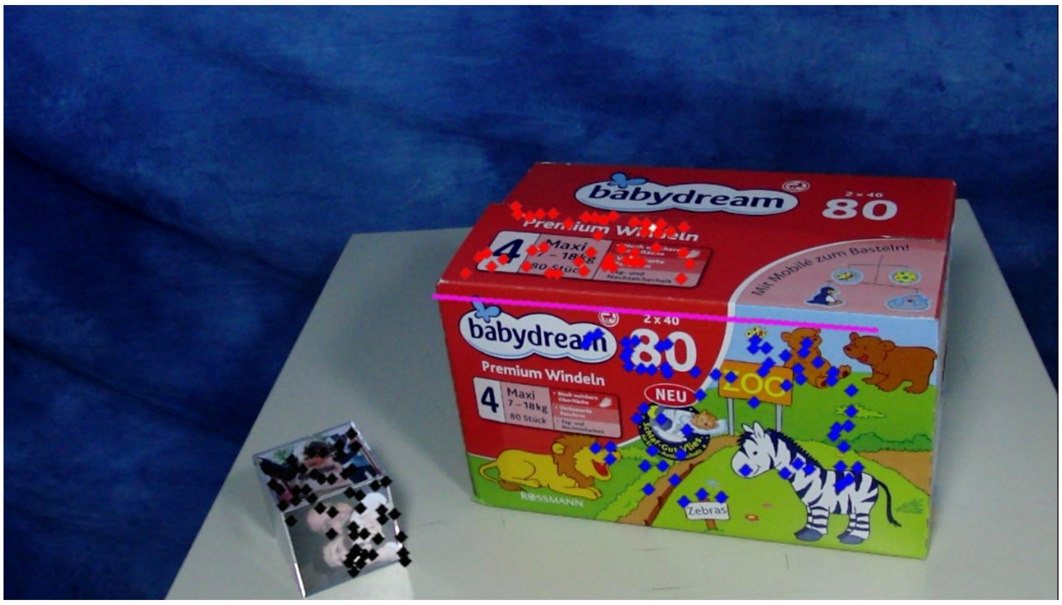
(a) Table Cluster

Results for Table Experiment				
Parts	Confidence scores			
	Rigid	Prismatic	Revolute	Disc
Background-Top Table	0.000	**0.917**	0.000	0.083
Background-Bottom Table	0.000	**0.995**	0.000	0.005
Top Table-Bottom Table	**0.690**	0.013	0.135	0.162

(b) Results

Figure 19: Experimental Results for the Table experiment. The table is moved from the left to the right, so that it looks like a prismatic motion. Two cluster on the moving table (red and blue) are compared to the background (not visible) and to each other. The algorithms finds the right results: The two clusters on the table are declared as rigidly connected. Also between each table cluster and the background two prismatic joints are found. Adapted from [10].

Appendix E: Diaper Box

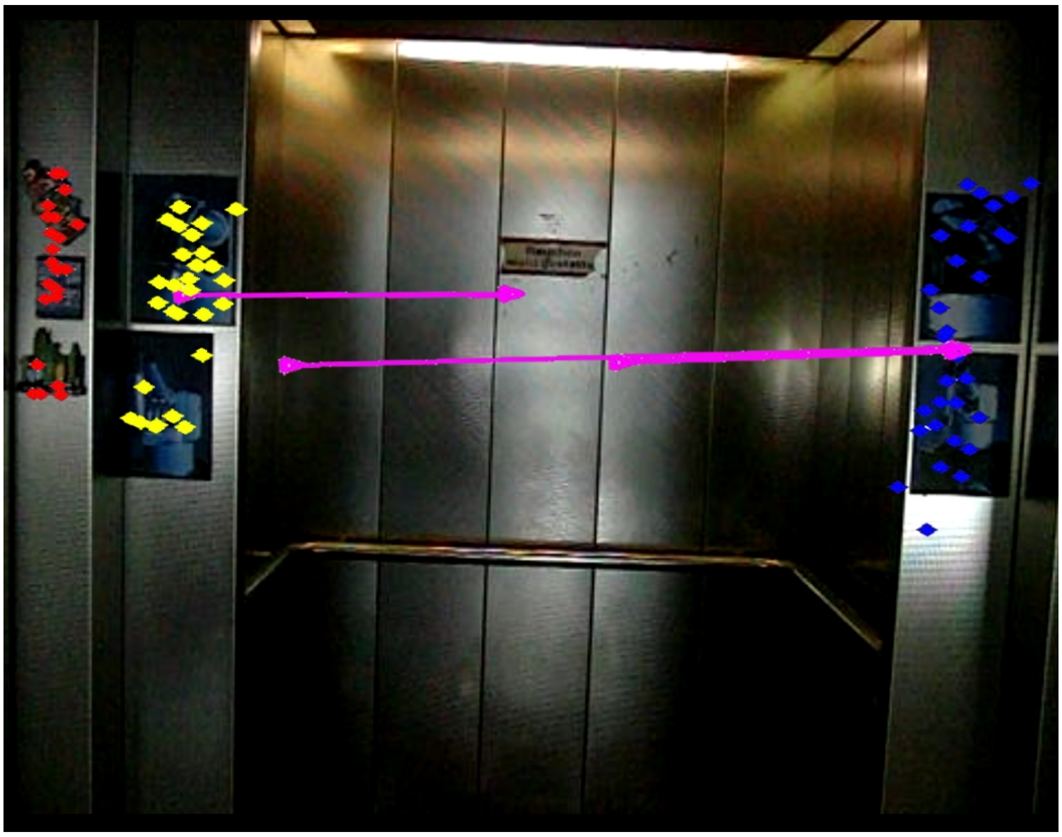

(a) Diaper Box Cluster

Results for Diaper Box Experiment				
Parts	Confidence scores			
	Rigid	Prismatic	Revolute	Disc
Background-Side	0.000	0.000	0.205	**0.794**
Background-Top	0.000	0.000	0.070	**0.930**
Side-Top	0.000	0.000	**0.968**	0.032

(b) Results

Figure 20: Experimental Results for the diaper box experiment. A background cluster on the photo cube (black), features on the side (blue) and the top (red) of the box can be seen. From the analysis, a revolute joint between the Side and the Top was detected. The position and orientation of the joint is given back from the algorithm. As also seen in the Laptop Experiment, features near to the axis of rotation do not exert much motion. Therefore they influence the position and orientation of the joint. Adapted from [10].

Appendix F: Elevator



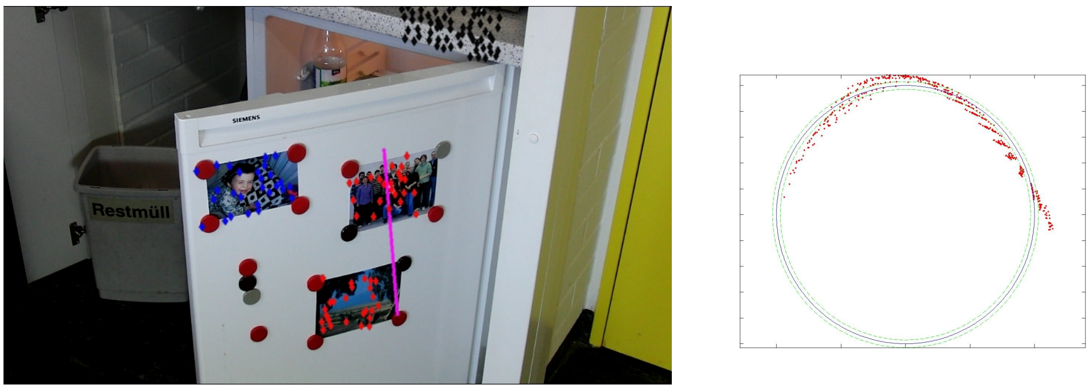
(a) Elevator Cluster

Results for Elevator Experiment				
Parts	Confidence scores			
	Rigid	Prismatic	Revolute	Disc
Left Door-Right Door	0.000	**0.995**	0.005	0.001
Left Door-Background	0.000	**0.994**	0.000	0.006
Background-Right Door	0.000	**0.955**	0.000	0.045

(b) Results

Figure 21: Experimental Results for the Elevator experiment. In the video, the doors are opening after the robot pushed the button on the door. Shown in the first figure is the last frame of the motion, with one static background cluster (red) and two clusters, one on each door (yellow and blue). The resulting axes between each cluster is plotted into the figure (violet). Note that the two axes between red-blue and yellow-blue are overlapping. Adapted from [10].

Appendix G: Fridge



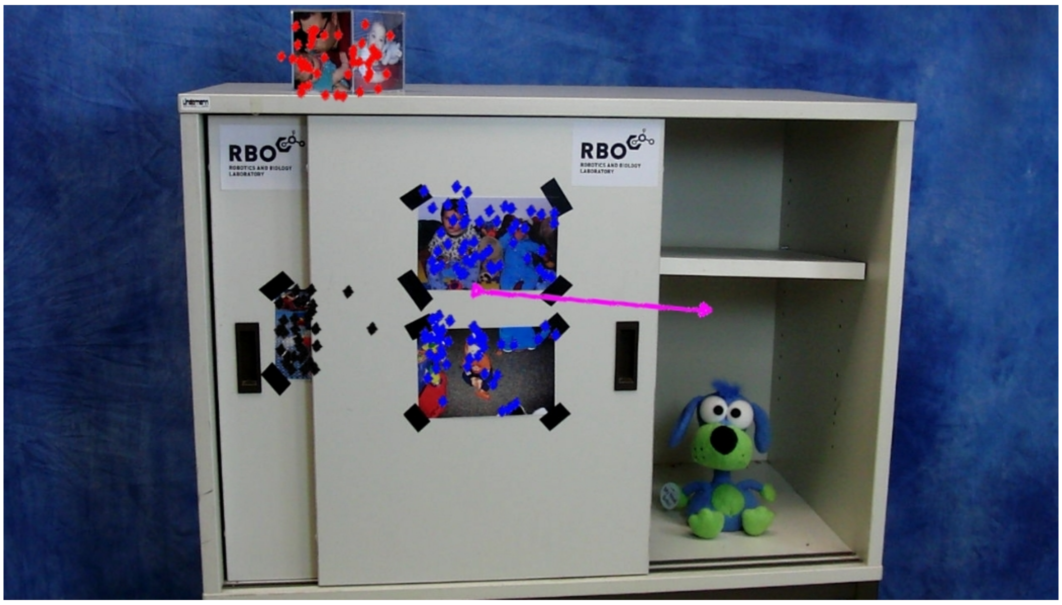
(a) Fridge Cluster

Results for Fridge Experiment				
Parts	Confidence scores			
	Rigid	Prismatic	Revolute	Disc
Background-Door1	0.000	0.000	0.278	**0.722**
Background-Door2	0.000	0.000	**0.893**	0.107
Door1-Door2	0.000	**0.919**	0.081	0.000

(b) Results

Figure 22: Experimental Results for the fridge experiment. Clusters on the background (black) and two on the door of the fridge (blue,red) can be seen. The algorithm fails to detect the right joint type between the red cluster and the background. As can be seen on the unit circle projection (see section 5.3 for details), the motion is not circular, but rather elliptical. Even if this is wrong, it shows that the algorithms works as intended. Only real circular motions are declared as revolute. While anything else will be discarded. The other joint is found by the algorithm, even if the position is not correct. Both cases indicate that the input data is very noisy and does not contain the exact 3d position of the features. Adapted from [10].

Appendix H: Sliding Door



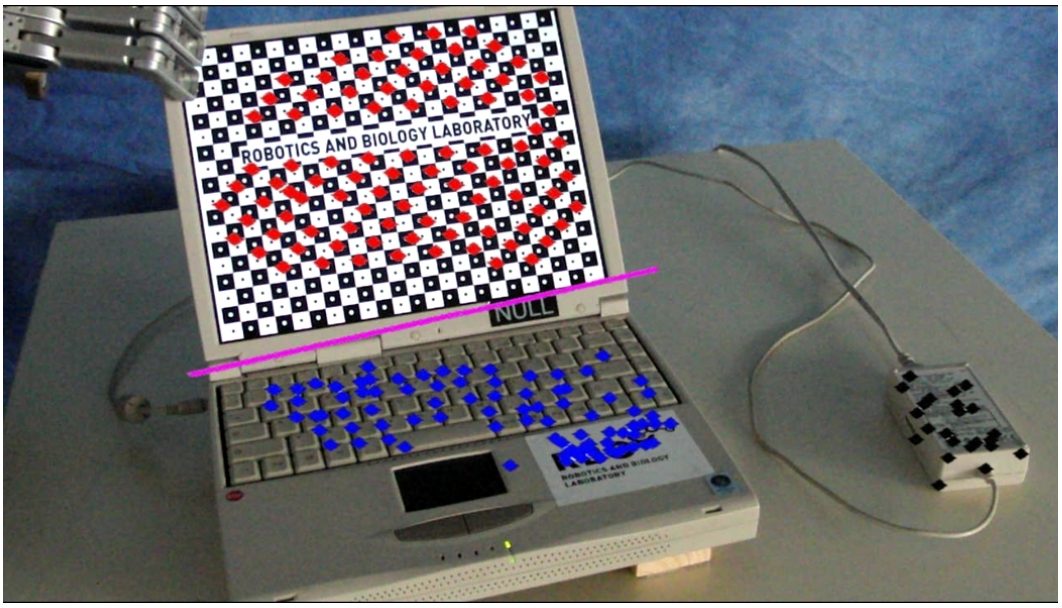
(a) Sliding Door Cluster

Results for Sliding Door Experiment				
Parts	Confidence scores			
	Rigid	Prismatic	Revolute	Disc
Top Part-Door	0.000	**0.997**	0.000	0.003
Top Part-Left Part	**1.000**	0.000	0.000	0.000
Door-Left Part	0.000	**0.997**	0.000	0.003

(b) Results

Figure 23: Experimental Results for the Sliding Door experiment. The door of the cabinet is opened during the interaction. One cluster on it (blue) is compared to the photo cube on the top (red) and one cluster on the other door (black). A rigid connection is detected between the two non moving clusters. The moving one is declared as prismatic to each of the static clusters. Both translation axes are shown in the figure. As one would expect, both are overlapping. Adapted from [10].

Appendix I: Laptop



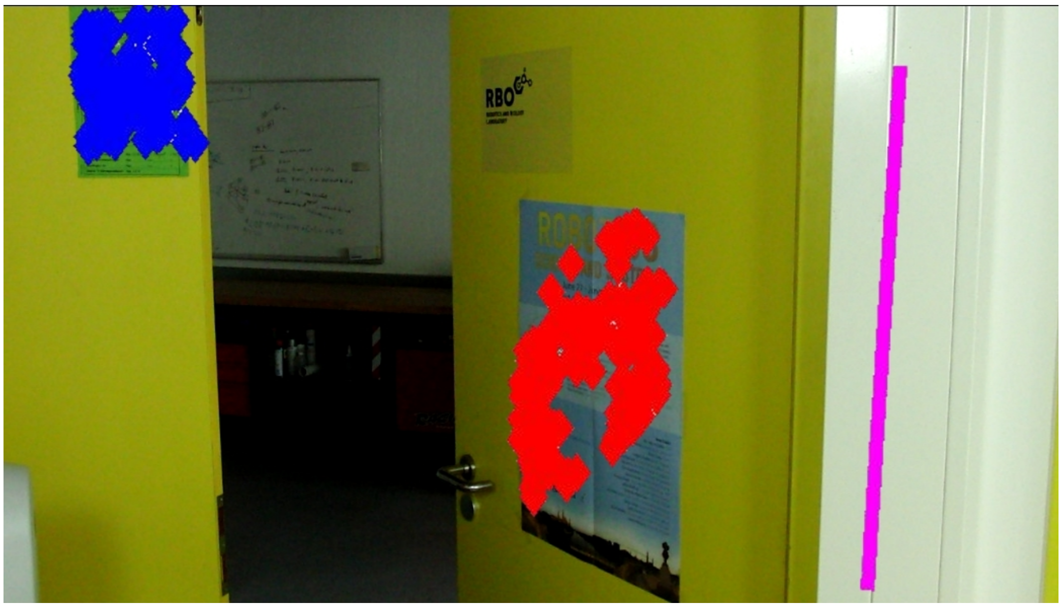
(a) Laptop Cluster

Results for Laptop Experiment				
Parts	Confidence scores			
	Rigid	Prismatic	Revolute	Disc
Power Supply-Keyboard	0.000	0.000	**0.703**	0.297
Power Supply-Screen	0.000	0.000	0.204	**0.796**
Keyboard-Screen	0.000	0.000	**0.946**	0.054

(b) Results

Figure 24: Experimental Results for the Laptop experiment. There are three clusters: the screen (red), the keyboard (blue) and a static background cluster, represented by features on the power supply (black). The resulting axes between keyboard and screen is shown in pink. Note that there are many features on the bottom part of the screen, which do not exert much motion. This is why the resulting axis got an orientation offset. Another axis between keyboard and power supply was detected by the algorithm. This is no flaw, but the laptop was moved in a circular motion over the table. Therefore, the algorithm believes that there is a revolute joint present. Further movement is obviously required to let the algorithm determine the true joint type between keyboard and power supply. To avoid confusion, this axis was not plotted into the image. Adapted from [10].

Appendix J: Door



(a) Door Cluster

Results for Door Experiment				
Parts	Confidence scores			
	Rigid	Prismatic	Revolute	Disc
Poster-Door	0.000	0.000	**1.000**	0.000

(b) Results

Figure 25: Experimental Results for the Door experiment. One cluster on a door (red) and one at the wall (blue) are compared to each other. The resulting axis of rotation has a small offset due to the noise of the motion. Adapted from [10].

[14]

References

[1] K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-d point sets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9:698–700, 1987.

[2] Bronstein, Ilja N., Semendjajew, Konstantin A., and Musiol, Gerhard. *Handbook of Mathematics*. Springer, August 2005.

[3] John J. Craig. Manipulator kinematics. In *Introduction to Robotics: Mechanics and Control*, chapter 3, pages 62–100. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, third edition, 2005.

[4] Anthony Dearden and Yiannis Demiris. Learning forward models for robots. In *in Proceedings of IJCAI*, pages 1440–1445, 2005.

[5] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24:381–395, June 1981.

[6] Berthold K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *J. Opt. Soc. Am. A*, 4(4):629–642, 1987.

[7] T. S. Huang, S. D. Blostein, and E. A. Margerum. Least-squares estimation of motion parameters from 3-d point correspondences. In *Proceedings IEEE Conference Computer Vision and Pattern Recognition*, pages 24–26, Miami Beach, FL, USA, 1986.

[8] Dov Katz and Oliver Brock. Extracting planar kinematic models using interactive perception. In *In Unifying Perspectives In Computational and Robot Vision*, volume 8 of *Lecture Notes in Electrical Engineering*, pages 11–23. Springer Verlag, May 2008.

[9] Dov Katz and Oliver Brock. Manipulating articulated objects with interactive perception. In *Proceedings of the International Conference on Advanced Robotics*, pages 272–277, Pasadena, USA, May 19-23 2008.

[10] Dov Katz, Andreas Orthey, and Oliver Brock. Interactive perception of articulated objects. In *12th International Symposium on Experimental Robotics*, Delhi, India, Dec 2010.

[11] A.G. Kirk, J.F. O'Brien, and D.A. Forsyth. Skeletal parameter estimation from optical motion capture data. volume 2, page 1185 vol. 2, jun. 2005.

[12] MathForum. Orthogonal distance regression planes. `http://mathforum.org/library/drmath/view/63765.html`, July 2003.

[13] Niloy J. Mitra, Natasha Gelfand, Helmut Pottmann, and Leonidas Guibas. Registration of point cloud data from a geometric optimization perspective. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, SGP '04, pages 22–31, New York, NY, USA, 2004. ACM.

[14] Orthey, Andreas. *3D Joint Detection*, January 2011.

[15] David Poole. Eigenvalues and eigenvectors. In *Linear Algebra Modern Introduction*, chapter 4, pages 252–362. Addison-Wesley, Reading, MA, second edition, 2006.

[16] David Poole. The singular value decomposition. In *Linear Algebra Modern Introduction*, chapter 7, pages 599–615. Addison-Wesley, Reading, MA, second edition, 2006.

[17] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical recipes in C (2nd ed.): the art of scientific computing.* Cambridge University Press, New York, NY, USA, 1992.

[18] David A. Ross, Daniel Tarlow, and Richard S. Zemel. Learning articulated structure and motion. *Int. J. Comput. Vision*, 88:214–237, June 2010.

[19] Jonathon Shlens. A Tutorial on Principal Component Analysis. Technical report, Systems Neurobiology Laboratory, Salk Insitute for Biological Studies, December 2005.

[20] J. Sturm, V. Pradeep, C. Stachniss, C. Plagemann, K. Konolige, and W. Burgard. Learning kinematic models for articulated objects. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2009.

[21] J. Sturm, C. Stachniss, V. Pradeep, C. Plagemann, K. Konolige, and W. Burgard. Towards understanding articulated objects. In *Proc. of the Workshop on Robot Manipulation at Robotics: Science and Systems Conference (RSS)*, 2009.

[22] Jürgen Sturm, Christian Plagemann, and Wolfram Burgard. Body schema learning for robotic manipulators from visual self-perception. *Journal of Physiology-Paris*, 103(3-5):220–231, 2009. Neurorobotics.

[23] Gabriel Taubin. Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13:1115–1138, November 1991.

[24] Leonid Taycher, John W. Fisher Iii, and Trevor Darrell. Recovering articulated model topology from observed motion. In *In In Proc. NIPS*, pages 1311–1318. MIT Press, 2002.

[25] Eric W. Weisstein. Point-line distance–3-dimensional. – from mathworld. `http://mathworld.wolfram.com/Point-LineDistance3-Dimensional.html`, July 2005.

[26] Wikipedia. Q-function — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/wiki/Q-function`, 2010. [Online; accessed 01-November-2010].

[27] Jingyu Yan and Marc Pollefeys. Automatic kinematic chain building from feature trajectories of articulated objects. In *CVPR (1)*, pages 712–719, 2006.