Probabilistic Depth Image Registration incorporating Nonvisual Information

Manuel Wüthrich, Peter Pastor, Ludovic Righetti, Aude Billard, Stefan Schaal

Abstract—In this paper, we derive a probabilistic registration algorithm for object modeling and tracking. In many robotics applications, such as manipulation tasks, nonvisual information about the movement of the object is available, which we will combine with the visual information. Furthermore we do not only consider observations of the object, but we also take space into account which has been observed to not be part of the object. Furthermore we are computing a posterior distribution over the relative alignment and not a point estimate as typically done in for example Iterative Closest Point (ICP). To our knowledge no existing algorithm meets these three conditions and we thus derive a novel registration algorithm in a Bayesian framework. Experimental results suggest that the proposed methods perform favorably in comparison to PCL implementations of feature mapping and ICP, especially if nonvisual information is available.

I. INTRODUCTION

In this paper we will focus on the scenario where the object has undergone between two point clouds. In our proposed method, we consider the acquisitions of these two images. Segmentation is not the focus of this work, we employ existing algorithms [1] to determine whether a pixel in the depth image belongs to the object or to the background.

A great deal of work has been done in this research area in the past years. In [2] an algorithm is presented which creates 3D models of objects while the camera or the object is moved. However, the point clouds have to be approximately aligned initially and the model is created off line by optimizing the alignment of all images simultaneously. Our method is more general in the sense that point clouds do not have to be approximately aligned. However, task-specific assumptions like that can be introduced to significantly reduce the computational time for finding the optimal alignment.

A lot of very promising work, such as [3], [4], has been published in the last years about scanning objects while they are being held by the robot. We however want to treat a more general case where we do not assume that the object is already grasped or can be grasped in a straightforward manner.

In [5] models are constructed by mapping shape primitives to the point clouds with promising results. In this work however we try to make as few assumptions as possible about the shape of the object and thus exclude the use of models or shape primitives.

Among the most popular algorithms that tackle the registration problem are Iterative Closest Point (ICP) and feature mapping algorithms and combinations of both [6], [7], [8], [9], [10]. We will compare the proposed method with these two approaches.

ICP has been proven to converge to a local minima [11]. In the scenario considered in this paper, an object can move very fast and therefore, point clouds of two subsequent images are not necessarily approximately aligned. This problem is usually tackled by initially aligning point clouds using a feature mapping algorithm [6], [8], [9]. These methods perform well, if different parts of the object can easily be distinguished. For objects with a homogeneous texture, color or local shape, feature matching can be problematic. Furthermore if the quality of the features degrades with the quality of the point cloud, noisy data can cause problems.

Often in robotics there is a great deal of nonvisual information about the transformation of the object available. In our scenario, this information can for example be that an object is pushed on a table and the movement will therefore be in a plane. If it is held by a robot, we approximately know how the object will move. This kind of information can certainly be incorporated in ICP and feature mapping algorithms, but they are not originally designed to do so.

ICP and feature mapping algorithms commonly optimize a cost function that is only dependent on the relative alignment between two point clouds. In our proposed method, we take into account the space which has been observed to not contain any part of the object. Our results suggest that taking this information into account leads to more robust registration results. Introducing visibility constraints has previously been shown to help in estimating the occluded shape of an unknown object [12].

Finally, feature mapping and ICP algorithms usually return a point estimate of the transformation and a fitness. It can however be preferable to have a more differentiated estimate of the transformation in form of a probability distribution over the 6 parameters of the transformation. This allows us to express, for example, that we are certain about the rotation around axis $x$ but uncertain about the translation in $y$ etc. In the results section we will show an example of the use of a
probability distribution as result.

To our knowledge, there is no registration algorithm that combines the three mentioned points:
1) Cost function based on visibility constraints.
2) Output of a posterior distribution over the estimated pose change.
3) Straightforward incorporation of task-relevant nonvisual information.

In the next section, we will derive the proposed registration algorithm in a Bayesian framework. In the result section, we show that under certain conditions that are quite common in the scenario of object model learning and tracking, our algorithm outperforms implementations of ICP and feature mapping methods.

II. DERIVATION

A. Incorporated Information

An overview of all the information we will make use of can be seen in Fig. 1. The input data \( D \) consists of the visual information \( V \) and the nonvisual information \( N \).

1) Nonvisual Information: In the context of a robotic manipulation task often a great deal of nonvisual information about the movement of an object is available. \( N \) can contain for example the information that the object will be moved on a table, that the robot has poked it with a certain movement or that the object is being held by the robot, and we thus know how it has moved approximately.

2) Visual Information: We divide the visual information into two types (see Fig. 2). Firstly there are surface patches which are observed by the depth camera, from now on referred to as patches \( P \). These patches can be represented as a point cloud and are thus the only information used by ICP and by most feature mapping algorithms.

There is however another very important piece of information. No part of the object is inside the green area in Fig. 2, this area defines thus a mask \( M \) for the object.

We will always register two depth images, \( A \) and \( B \), at a time, therefore we have of course the masks, \( M_A \) and \( M_B \), as well as the patches, \( P_A \) and \( P_B \), from each image (see Fig. 1).

B. Parametrization

1) Coordinate system: Given that we will work with depth images we choose a suitable parametrization assuming the pinhole model for the camera. The first two parameters, \( w \) and \( h \), are chosen to be the projections of a 3D point onto a virtual image plane given a focal length of 1m, see Fig. 3. The third parameter \( r \) is the depth of the 3D point. These coordinates will be called ray coordinates. They are derived from Cartesian coordinates as follows:

\[
w = \frac{x}{z},\quad h = \frac{y}{z},\quad r = \sqrt{x^2 + y^2 + z^2}
\]

2) Rigid body transformation: The rigid body transformation \( T \) has six independent parameters \( T = (T_1,...,T_6)\). The parametrization can be chosen to be whatever is convenient for a given application.

C. Measurement Error

Due to measurement errors in the camera, an observed 3D point \( p \) will not exactly correspond to the true point \( s \) on the object surface. As a measurement model \( p(s) \) we use a normal distribution in ray coordinates.

\[
p(p|s) = \mathcal{N}(s|p, L)
\]

The covariance matrix \( L \) is camera specific. The only assumption we make in our derivation is that the covariance matrix is such that \( p(p|s) \) can be reasonably well approximated as being constant within a pixel. This assumption is sensible because the depth camera is not able to distinguish between points within the range of one pixel. Furthermore, assuming that \( p(p) \) and \( p(s) \) are uniform in the range of the depth camera, we have \( p(s|p) = p(p|s) \).

D. Derivation

Our objective is to express \( p(T|D) \), the probability distribution over the transformation \( T \) the object has undergone, given all the available data \( D \). Applying Bayes we have

\[
p(T|D) = \frac{p(D|N,P,T)p(T|N,P)}{\int p(D|N,P,T)p(T|N,P)dT}
\]

Fig. 3. Schematic representation of eight pixels acquired by the depth camera. The coordinates \( r \) and \( w \) are represented, while \( h \) would be perpendicular to the image plane.
In \( p(M|N, P, T) \), given the transformation \( T \), the mask \( M \) does not depend on the nonvisual information \( N \) and we thus have

\[
p(M|N, P, T) = p(M|P, T) = p(M_A, M_B|P_A, P_B, T)
\]

We assume \( M_A \) and \( M_B \) to be independent because the mask observed in one image does not give us any useful information about the mask observed in the other image.

\[
p(M|N, P, T) = p(M_A|P_A, P_B, T)p(M_B|P_A, P_B, T)
\]

As \( M_A \) and \( P_A \) are from the same image, the object \( P_A \) will necessarily be respected in the mask \( M_A \). \( P_A \) does thus not add any information to the first term and can be removed. Similarly, for the second term we can omit \( P_B \).

\[
p(M|N, P, T) = p(M_A|P_B, T)p(M_B|P_A, T)
\]

It is reasonable to assume that the priors \( p(M|T) \) and \( p(P|T) \) are uniform because we do not have any prior information about the distribution of the points and the mask. Applying of Bayes’ rule, we thus have

\[
p(M|N, P, T) = k p(P_B|M_A, T)p(P_A|M_B, T)
\]

with \( k \) being a constant.

Inserting this result into Eq. 4 we obtain

\[
p(T|D) = \frac{p(P_B|M_A, T)p(P_A|M_B, T)p(T|N, P)}{\int p(P_B|M_A, T)p(P_A|M_B, T)p(T|N, P)dT}
\]

Finding this distribution is intractable, but for most purposes we do not need the distribution itself, we only use it for evaluating expectations. We thus need to find the expectation of a function \( f(T) \) expressing a property of \( T \) required for a given application. If \( f \) is for example identity \( (f(T) = T) \), then \( E(f(T)) = E(T) \), or if \( f(T) = (T - E(T))'(T - E(T)) \) then \( E(f(T)) \) is the covariance matrix of \( T \). The expectation of a function of \( T \) is

\[
E(f(T)) = \frac{\int p(P_B|M_A, T)p(P_A|M_B, T)p(T|N, P)f(T)dT}{\int p(P_B|M_A, T)p(P_A|M_B, T)p(T|N, P)dT}
\]

\[
E(f(T)) \approx \sum_{l=1}^{L} w^{(l)} f(T^{(l)})
\]

Where the samples \( T^{(l)} \) are drawn from \( p(T|N, P) \). The sampling weights \( w^{(l)} \) are defined by

\[
w^{(l)} = \frac{p(P_B|M_A, T^{(l)})p(P_A|M_B, T^{(l)})}{\sum_{m=1}^{L} p(P_B|M_A, T^{(m)})p(P_A|M_B, T^{(m)})}
\]

We thus have represented \( p(T|D) \) by a set of samples \( \{T^{(l)}\} \) and the corresponding weights \( \{w^{(l)}\} \). The samples are drawn from \( p(T|N, P) \), in other words, we will create a distribution, from which it is possible to sample, taking into account the nonvisual information as well as the observed surface patches. This distribution will be defined indepen-

The terms \( p(P_A|M_B, T) \) and \( p(P_B|M_A, T) \) determine the weight of a given sample. The first one expresses the likelihood of \( T \) given the patches observed in \( A \) and the mask observed in \( B \). It essentially states that the transformation \( T \) has to be such that the patches observed in \( A \) fit into the mask observed in \( B \). Conversely the second term assures that the patches from \( B \) fit into the mask from \( A \).

Now we will express \( p(P_A|M_B, T) \). \( P_A \) is the set of all the surface patches observed in image \( A \) and can be expressed as a set of points \( \{a_1, a_2, ..., a_n\} \). Similarly we have \( P_B = \{b_1, b_2, ..., b_m\} \). We can now write

\[
p(P_B|M_A, T) = p(b_1, b_2, ..., b_m|M_A, T)
\]

Given the mask \( M_A \) observed in image \( A \), we look at the points \( P_B \) observed in image \( B \) as independent observations:

\[
p(P_B|M_A, T) = \prod_{j=1}^{m} p(b_j|M_A, T)
\]

After the derivation in Appx. A we have

\[
p(b_j|M_A, T) \propto K_2 \sum_{i=1}^{n} e^{-\frac{1}{2}(b_j - a_i)^{T} D (b_j - a_i)}
\]

\[
(1 + erf(v^{T}(b_j - a_i)))
\]

with \( D, v, K_2, Z \) defined in Appx. A. The second term in Eq. 12, \( p(P_A|M_B, T) \), can be expressed analogously.

E. Discussion

The first term in Eq. 15 is a Gaussian over the parameters \( w, h \) with mean \((w_i, h_i)^{T}\). This term accounts for the fact that the closer \( b_j \) is to a pixel \( i \), the likelier it is that the point which has been observed at \( b_j \) in image \( B \) is observed in pixel \( i \) in image \( A \). The second term goes to zero if the depth of \( b_j \) is smaller than the depth at the pixel where it is projected on in image \( A \), which is necessary in order to respect the mask \( M_A \).

Given that \( p(P_B|M_A, T) \) (see Eq. 14) is the product of all

![Fig. 4. Schematic representation of \( p(b_j|M_A, T) \) (blue), \( p(a_i|M_B, T) \) (red), \( \{a_1, a_2, ..., a_n\} \) (blue dots) and \( \{b_1, b_2, ..., b_m\} \) (red dots) \( p(b_j|M_A, T) \), it is zero if any \( p(b_j|M_A, T) \) is zero. This result is illustrated in Fig. 4, all of the red points have to be inside the blue area.](image)
III. IMPLEMENTATION

The only parameter that has to be determined for our algorithm is the covariance matrix of the camera uncertainty (Eq. 2). This is however not a parameter that has to be optimized, it represents a meaningful quantity and should be estimated for the depth camera that is used. For our experiments with the Kinect camera we estimated the covariance matrix to be isotropic with \( \sigma = 0.002 \), which corresponds approximately to the resolution in ray coordinates of the Kinect. These values are a very rough estimation of the properties of the Kinect, but they prove to work well in the experiments.

The core of the algorithm looks as follows:

- For \( K \) samples
  - Sample from \( p(T|N,P) \)
  - For all points in \( B \)
    * if \( p(b_j|M_A,T) \) is zero, sample a new transform
    * else \( p(P_B|M_A,T^{(l)}) = p(b_j|M_A,T) \)
  - Do the same for points in \( A \)
- Given all the \( p(P_A|M_B,T^{(l)}) \) and \( p(P_B|M_A,T^{(l)}) \) we can compute the covariance matrix and the mean of \( T \) according to Eq. 11 and Eq. 12.

IV. RESULTS

As mentioned in the introduction, the algorithms we want to compare against are ICP and feature mapping. We used the implementations in the Point Cloud Library (PCL) of these algorithms for our evaluation. We employed FPFH features which are described in [10].

Our dataset consists of three objects, a box, a flashlight and a tube. Our dataset is small, the three objects however have a big variety in shape as seen in Fig. 5, and therefore this evaluation gives us a reasonable idea about the performance of our algorithm. Admittedly a broader evaluation will be necessary for a more precise assessment of the performance. In the associated video the algorithm is applied to a series of different objects on a tabletop.

Each of the three objects has been rotated in steps of about 25° and translated by a few cm 14 times on a tabletop. At each step we acquire a depth image and measure the object's exact position and orientation which will be used as ground truth. For evaluation we will align each image to the next, which gives a total of 13 alignments per object.

We compare our algorithm to ICP, feature mapping and feature mapping with subsequent ICP. We use the implementations of these algorithms in the Point Cloud Library (PCL) [1]. The feature mapping algorithm uses Fast Point Feature Histograms (FPFH) as shape features [10]. We used these algorithms to our best knowledge and implemented them as suggested in tutorials of PCL. We do not claim that the performance we measure here for ICP and feature mapping is the maximum that can be achieved with these algorithms, but it serves as a good point of comparison for our new algorithm.

A. Evaluation of Alignment Performance without Nonvisual Information

In order to obtain a general estimate of the alignment performance of our algorithm we only make very general assumptions for the sampling distribution \( p(T|N,P_A,P_B) \). We will assume that we have no information \( N \), we do thus not use the information that object has only been translated and rotated on a table top. We only assume that the center of mass of \( P_A \) will be no further than 4cm from the center of mass of \( P_B \) in the aligned images and that the object will not be rotated by more than 50 degrees at a time. Note that these assumptions leave a very big search space open, and, therefore, we have to draw a very large amount of samples – about 100 million – and the algorithm is thus slow and takes about 30 seconds per image. ICP took about 1 second and feature mapping took about 5. In practice however we will have much stronger sampling distributions which will accelerate our algorithm considerably.

In Fig. 6 we present the box-plots of the alignment error in degree of the four algorithms for all the objects. Our algorithm performs favorably compared to these implementations of ICP and feature mapping. We will now try to investigate how this advantage emerges.

Fig. 7 shows an alignment performed by ICP with an error of 32°. The top image shows the aligned point clouds. The two bottom images represent the information about the mask. The left one illustrates \( p(b_j|M_A,T) \). In the blue area the object has been observed, in the gray area background has been observed, and in the black area no observation has been
made. The red points represent the points observed in \( B \) projected into image \( A \), \( [b_j]_A \). The result of our derivation suggests that the red points can only be in the blue or black area. If a point \( [b_j]_A \) is located on a pixel \( a_j \) in the blue area its distance to the camera \( r \) has to be approximately equal or larger than the depth measured at \( a_j \). If the point is located in the black area its distance to the camera can be arbitrary because no depth has been measured at the corresponding pixel.

ICP however only uses the information contained in the point clouds, which are quite sparse in the considered images. Looking at the top image of Fig. 7 it does not surprise that ICP performs poorly on this data. If we look at the two bottom images however we can see that many of the projected points are in front of the background. Taking this information into account we thus know that this alignment is not correct. In Fig. 8 the alignment of the same two images by our algorithm is shown. Even though the point clouds are quite sparse it has performed well thanks to the information about the mask.

Fig. 9 illustrates a problem of a different nature that occurred with feature mapping and ICP. The problem here is that this box, looking only at the point clouds, allows different alignments. The red point cloud should be rotated about 90\(^\circ\) to the left. The box fortunately is a little bit broader than wide. Taking the mask into account we can thus resolve this ambiguity. In the small image on the right on the bottom we see that many blue points are in front of the background which enables our algorithm to discard this alignment. Our algorithm aligned these images with an error of 2.7\(^\circ\).

These results illustrate that taking the mask into account can resolve important problems.

B. Evaluation of Alignment Performance with Nonvisual Information

Now we will show the benefits of taking nonvisual information \( N \) about the transformation into account. The dataset we have been working on consists of translations and rotations on a tabletop. Before we did not make use of this information. Now we include this information in the sampling distribution of our algorithm. We thus only sample from translation and rotations in the plane of the table. Of course our search space is much smaller now, and therefore we need less samples. The computational time is reduced to about 0.5 seconds per alignment. The additional information of course also contributes to the alignment performance as we can see in Fig. 11. There may be ways to make use of nonvisual information in ICP and feature mapping algorithms as well, it does however not emerge naturally and we did not try to do so. In Fig. 11 two very sparse point clouds are shown. Even for a human it is hard to tell how these should be aligned. ICP, feature mapping and feature mapping with subsequent ICP all aligned these point clouds with an error of at least 51\(^\circ\). Our algorithm without the table prior produced an error of 11\(^\circ\). With the prior however these points are aligned with an error of only 6.3\(^\circ\). This example
Fig. 10. Boxplot of alignment error for different algorithms.

Fig. 11. Very sparse point clouds of flashlight aligned with help of nonvisual information. Alignment error = 6.3°.

Fig. 12. The transformations on the right have been optimized, on the left not.

C. Evaluation of Alignment Performance with Loop-closure

We argued that as output of the alignment we prefer a probability distribution to a point estimate. As an example why this is useful we will merge all the point clouds of the box, aligned by our algorithm with the table prior, into one point cloud, as shown in Fig. 12. Frame 1 has been aligned to image 2, image 2 to image 3 and so on. Between the first and the last image the object has been rotated by around 360°, we can thus close the loop and align the last to the first image. Therefore we now have redundant information about the transformation of each image, and can optimize these transformations. This optimization is problematic if we only have a point estimate of each transformation. Fortunately however we can compute the mean as well as the covariance matrix of each transformation. We can thus estimate the probability of each transformation assuming that its distribution is normal with covariance matrix and mean as computed by our algorithm. We numerically maximize the joint probability of all the transformations using the graph optimization algorithm described in [13]. The difference between the transformations which are optimized and the ones which are not is illustrated in Fig. 12. This illustrates one of the advantages of having a posterior distribution rather than a point estimate.

V. CONCLUSION AND FUTURE WORK

The results of our evaluation are promising, but for a full assessment of the performance of our novel algorithm many more experiments are necessary. The derivation of this algorithm is general and does not assume in any way that the object is on top of a table. Note that we only used this information where we explicitly mentioned it. It might however be favorable for the performance of our algorithm because the depth camera always manages to measure the depth on pixels which are on the table top. This gives us a lot of information about the mask. The next step will be to measure the performance of the algorithm in other cases, such as when the object is held by the robot hand.

The core of our algorithm is sampling, it can thus easily be parallelized or even implemented on a GPU in order to reduce the computational time.

In our sampling distribution $p(T|N, P_A, P_B)$ we have barely used the information coming from $P_A, P_B$. This information is not very important if we already have a good idea how the object has moved given $N$. If we have however no nonvisual information about how the object has moved we can make assumptions based on $P_A, P_B$. These assumptions are specific for a given application. If we know for example that we will observe only objects that are much longer in one dimension than in the others, then we can assume that the first Principal Component of $A$ is approximately aligned with the first Principal Component of $B$. Another possibility is to employ features. If we compute features for each point in $A$ and $B$ we can create a set of possible matches. Then we can sample from these matches, three at a time, which
gives us a sample for $T$. We might however inherit problems of feature mapping algorithms.

When we ran the algorithm on the robot we moved its arms manually. This was of course only for evaluation, a possible application of the algorithm is to be used in the context of manipulation tasks. There are numerous possibilities, such as using the algorithm in a grasping pipeline. If the robot encounters, for example, an object which does not have an obvious associated grasp observing it from only one side, we can start poking it with actions that minimize the uncertainty in the alignment. While the object is being moved around, our algorithm tracks it and completes a model. The more information we gain, the more likely are we to select the correct grasp.

In summary, we can say that there are many applications and possible extensions for this algorithm. Its most important feature is that due to its general formulation, it can make use of all the information available in a given case.

REFERENCES

[1] R. B. Rusu and S. Cousins, “3D is here: Point Cloud Library (PCL),” in The IEEE International Conference on Robotics and Automation (ICRA), 2011.

[2] Y. Cui, S. Schuon, C. Derek, S. Thrun, and C. Theobalt, “3D Shape Scanning with a Time-of-Flight Camera,” Proc. IEEE Computer Vision and Pattern Recognition, 2010.

[3] M. Krainin, P. Henry, X. Ren, and D. Fox, “Manipulator and Object Tracking for 3D Object Modeling,” International Journal of Robotics Research, 2011.

[4] W. Ho Li and L. Kleeman, “Interactive Learning of Visually Symmetric Objects,” in Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems, 2009, pp. 4751–4756.

[5] R. B. Rusu, N. Blodow, Z. C. Marton, and M. Beetz, “Close-range Scene Segmentation and Reconstruction of 3D Point Cloud Maps for Mobile Manipulation in Domestic Environments,” in Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems, 2009.

[6] J. Dai and J. Yang, “A Novel Two-Stage Algorithm for Accurate Registration of 3-D Point Clouds,” in Proc. Intl Conf. on Multimedia Technology (ICMT), 2011, pp. 6187 – 6191.

[7] Y. Liu and M. A. Rodrigues, “Accurate Registration of Structured Data using Two Overlapping Range Images,” in Proceedings of the IEEE Intl Conf. on Robotics and Automation, 2002, pp. 2519–2524.

[8] R. Bogdan Rusu, N. Blodow, Z. C. Marton, and M. Beetz, “Aligning Point Cloud Views using Persistent Feature Histograms,” in Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems, 2008.

[9] H. Fukai and G. Xu, “Fast and robust registration of multiple 3d point clouds,” in Intl Symp. on Robot and Human Interactive Communication (RO-MAN), 2011, pp. 331 – 336.

[10] R. B. Rusu, N. Blodow, and M. Beetz, “Fast point feature histograms (fpfh) for 3d registration,” in The IEEE International Conference on Robotics and Automation (ICRA), 2009.

[11] P. J. Besl and N. D. McKay, “A method for registration of 3-d shapes,” IEEE Trans. on Pattern Analysis and Machine Intelligence (PAMI), vol. 14, pp. 239–256, 1992.

[12] J. Bohg, M. Johnson-Roberson, B. León, J. Felip, X. Gratat, N. Bergström, D. Kragic, and A. Morales, “Mind the Gap - Robotic Grasping under Incomplete Observation,” in IEEE International Conference on Robotics and Automation (ICRA), May 2011.

[13] G. Grisetti, R. Kuemmerle, H. Strasdat, and K. Konolige, “g2o: A general framework for graph optimization,” in The IEEE International Conference on Robotics and Automation (ICRA), 2011.
A. Derivation of \( p(b|M_A, T) \)

As explained in Appx. B, the transformation from ray coordinates in image A to ray coordinates in image B is not linear, it can however be approximated linearly around a point. Point \( b \) is expressed in ray coordinates in image B and \( a \) is expressed in ray coordinates in image A.

\[
p(b|M_A, T) = \int_{-\infty}^{\infty} p(b|a, T)p(a|M_A)da
\]

(16)

\( p(b|a, T) \) expresses the probability distribution over the position of a point \( b \) observed in B given that we have observed the same point at \( a \) in A. With \( s \) being the underlying point, expressed in ray coordinates in image A, we can write

\[
p(b|a, T) = \int_{-\infty}^{\infty} p(s|T)p(s,a,T)ds \quad \text{and inserting Eq. 2 we obtain} \quad = \int_{-\infty}^{\infty} N(b|s_B, L)N(s|a, L)ds
\]

(17)

Given that \( [s]_B \) is only relevant in the neighborhood of \( b \) we can replace \( [s]_B \) by its linear approximation around \( b \) obtained in Appx. B:

\[
p(b|a, T) = \int_{-\infty}^{\infty} N(b + Q_B R Q_A^{-1}(s - [b]_A), L)N(s|a, L)ds
\]

(18)

\[
p(b|a, T) = K_1 e^{-\frac{1}{2}(a-[b]_A)^T \Lambda(a-[b]_A)}
\]

(19)

\[
K_1 = \frac{1}{(2\pi)^{3/2}|L + Q_B R Q_A^{-1} L Q_A^{-1} R^T Q_B^{-1}|^{1/2}}.
\]

As explained in the assumptions section, the whole term inside the integral of Eq. 16 can be approximated as being constant within the range of a pixel. The integral over \( w \) and \( h \) thus becomes a sum over the number of pixels \( n \):

\[
p(b|M_A, T) \propto \sum_{i=1}^{n} \int_{-\infty}^{\infty} p(b|w_i, h_i, r) p(w_i, h_i, r|M_A)dr
\]

(21)

\( p(w_i, h_i, r|M_A) \) is the probability distribution over the observation of \( b \) in A, given the mask \( M_A \). This probability distribution is equal to zero in the green area of Fig. 2 because we know that no part of the object has been observed there. Everywhere else it is uniform because we have no further information, considering only the mask. The green area, for a pixel \((w_i, h_i)\), corresponds to the range between the camera and the depth measured at the aforesaid pixel \((r_i)\). Therefore the probability distribution is equal to zero for \( r < r_i \) and uniform for \( r \geq r_i \). This can easily be translated into limits for the integral, and we obtain

\[
p(b|M_A, T) \propto \sum_{i=1}^{n} \int_{r_i}^{\infty} p(b|w_i, h_i, r)dr
\]

(22)

We can now integrate and obtain

\[
p(b|M_A, T) \propto K_2 \sum_{i=1}^{n} e^{-\frac{1}{2}([b]_A - a_i)^T Q_A^{-1} ([b]_A - a_i)} (1 + erf(v^T([b]_A - a_i)))
\]

(23)

\[
\Lambda^{-1} = Q_A R^{-1} Q_B^{-1} L Q_A^{-1} R^T Q_B^{-1} + L,
\]

\[
K_2 = \frac{1}{|L + Q_B R Q_A^{-1} L Q_A^{-1} R^T Q_B^{-1}|^{1/2}}.
\]

(24)

\[
D = \frac{1}{\Lambda_{33}} \begin{bmatrix} \Lambda_{11} \Lambda_{33} - \Lambda_{31}^2 & \Lambda_{33} \Lambda_{21} - \Lambda_{31} \Lambda_{32} \\ \Lambda_{33} \Lambda_{21} - \Lambda_{31} \Lambda_{32} & \Lambda_{22} \Lambda_{33} - \Lambda_{32}^2 \end{bmatrix}, \quad v = \frac{1}{\sqrt{2 \Lambda_{33}}} \begin{bmatrix} \Lambda_{31} \\ \Lambda_{32} \\ \Lambda_{33} \end{bmatrix}
\]

(25)

B. Linear approximation to ray coordinate transformation

We want to linearly approximate the transformation from ray coordinates in image A, to ray coordinates in image B, around the point \( b \), which is defined in ray coordinates in image B. With \( s \) defined in image A, it is straightforward to show that

\[
[s]_B \approx b + Q_B R Q_A^{-1}(s - [b]_A)
\]

with \( Q_A = \begin{bmatrix} \frac{\partial w}{\partial x} & \frac{\partial w}{\partial y} & \frac{\partial w}{\partial z} \\ \frac{\partial y}{\partial x} & \frac{\partial y}{\partial y} & \frac{\partial y}{\partial z} \\ \frac{\partial z}{\partial x} & \frac{\partial z}{\partial y} & \frac{\partial z}{\partial z} \end{bmatrix} \) and \( Q_B = \begin{bmatrix} \frac{\partial w}{\partial x} & \frac{\partial w}{\partial y} & \frac{\partial w}{\partial z} \\ \frac{\partial y}{\partial x} & \frac{\partial y}{\partial y} & \frac{\partial y}{\partial z} \\ \frac{\partial z}{\partial x} & \frac{\partial z}{\partial y} & \frac{\partial z}{\partial z} \end{bmatrix} \)