Robot Arm Pose Estimation by Pixel-wise Regression of Joint Angles

Felix Widmaier[1,2], Daniel Kappler[1], Stefan Schaal[1,3], Jeannette Bohg[1]

***Abstract*— To achieve accurate vision-based control with a robotic arm, a good hand-eye coordination is required. However, knowing the current configuration of the arm can be very difficult due to noisy readings from joint encoders or an inaccurate hand-eye calibration. We propose an approach for robot arm pose estimation that uses depth images of the arm as input to directly estimate angular joint positions. This is a frame-by-frame method which does not rely on good initialisation of the solution from the previous frames or knowledge from the joint encoders. For estimation, we employ a random regression forest which is trained on synthetically generated data. We compare different training objectives of the forest and also analyse the influence of prior segmentation of the arms on accuracy. We show that this approach improves previous work both in terms of computational complexity and accuracy. Despite being trained on synthetic data only, we demonstrate that the estimation also works on real depth images.**

I. Introduction

For autonomous, robotic grasping and manipulation, knowing the current pose of the robot's manipulator is important to achieve a good hand-eye coordination. Given the kinematics of the arm and the current joint angles, the pose of each link relative to the camera can be computed. However, getting good estimates for these angles can be difficult as position encoders may have considerable inaccuracies depending on the robot. An example for this is the ARM robot [1], which has encoders in the motors which are located in the shoulders but not directly at the joints. Estimation of the joint positions is thus prone to inaccuracies due to variable cable stretch (see Fig. 1). Therefore, additional techniques for improving the estimated arm pose are necessary.

This paper presents an approach of frame-by-frame joint angle estimation using depth images as input. It does not require initialization of the solution from e.g. previous frames or joint encoder readings. It is therefore also not prone to an bad initial guess and can immediately recover when losing track of the arm in some frames. Estimation is done with a *Random Forest* (RF) which is trained on synthetically rendered depth images from which large annotated training sets can be easily built. We analyse the influence of different training objectives on the accuracy of the RF. Apart from the standard *mean squared error* (MSE) training objective, we also use the specialized *DISP distance* [2]. This is a model-dependent metric for rigid and articulated body displacement that measures distance in configuration space. Moreover, we show that further improvement can be gained by using a second RF for prior foreground segmentation of the images.

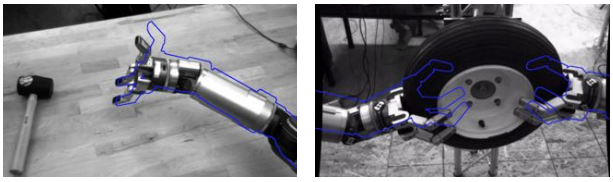


Fig. 1. Visualization of the encoder error. The blue outline shows the estimated pose of the arm based on the encoder readings. The considerable error between estimated and real arm pose makes any fine manipulation task very challenging. The error increases if there is an additional load applied at the end-effector.

The approach presents an improvement of our previous work [3] where we use an RF optimized on the proxy-objective of pixel-wise part classification. By retrofitting the leaf nodes of the classification forest, 3D joint positions can be estimated in the test images. These then serve as the basis to compute the final kinematically feasible robot arm pose. The approach presented in this paper simplifies the estimation by directly training the RF on the desired joint angles instead of part labels.

This reduces the number of intermediate steps as instead of collecting and clustering position votes for all joints to iteratively find an optimal solution for the arm configuration that matches the position estimates, we directly collect arm configuration votes which only have to be combined by a confidence-weighted mean. In the experimental section, we show that this also improves the estimation accuracy.

A. Related Work

Robust, accurate and real-time visual tracking of the robot arm is a long-standing problem. Often markers are attached to the arm to simplify detection [4], [5]. One has to take care, however, that these markers are always in sight of the camera which may constraint the possible arm poses. Another approach is to use a 3D model of the robot to detect the arm in the image without the need for markers [6]–[9]. These are often variants of *Iterative Closest Points* (ICP), i.e. local optimization methods that require a good initialization.

RFs [10] have been successfully applied to a wide range of vision problems [11]–[17]. They can naturally handle multi-class problems and are comparatively fast which makes them suitable for real-time applications. In [3] a classification RF is used for robot arm pose estimation based on single depth images. The forest is trained for pixel-wise classification of robot arms parts. By retrofitting the leaf nodes, they contain votes for relative 3D joint axis positions. At test

[1]Autonomous Motion Department at the Max-Planck-Institute for Intelligent Systems, Tübingen, Germany

[2]Student at the Eberhard Karls University of Tübingen, Germany

[3]Computational Learning and Motor Control lab at the University of Southern California, Los Angeles, CA, USA

time, these votes are cast per pixel. The 3D joint axis positions are estimated by clustering in the voting space. Given these, a virtual arm is optimally aligned with these positions through inverse kinematics to yield the final joint angles. In [18] a similar approach to estimate the pose of various articulated models is proposed. The authors train an RF to estimate object part labels. Then they sample a set of joint configurations given the retrofitted leafs of the RF and evaluate how well they fit the depth image. The best hypothesis is then refined. These refinement may be more accurate but also computationally more expensive compared to the approach in in [3], as the whole depth image is considered instead of only a few 3D joint axes positions.

In this paper, we improve our previous approach [3] by training a regression RF directly on the arm poses. This requires a metric to measure the distance between different poses. Defining such a metric is non-trivial as translation and rotation have different units and are therefore not directly comparable. This can be solved by using metrics which take the model of the object into account [2], [19]–[23]. In this paper, we compare the performance of such a model-based metric (the *DISP distance* [2]) to the simple Euclidean distance in the configuration space of the arm. While the former takes the geometry and kinematics of the robot arm into account, the latter is agnostic to the geometry of the problem.

B. Main Contribution

We present an approach for robot arm pose estimation that uses an RF to directly regress to joint angles instead of using a proxy objective such as part-based classification. As opposed to the dominant approach taken in related work, the method uses single depth images and does not need any initialization or knowledge about former frames. It outperforms our previous work [3] both in terms of accuracy and efficiency. We also analyze the influence of different objective functions for the RF-training on performance of the approach. While the proposed method works on unpreprocessed depth images, we show that prior segmentation using a second RF can be beneficial to further increase accuracy.

II. Problem Description and Foundations

We want to estimate a function $f(I)$ that takes a depth image I of the robot arm as input and returns a vector $\boldsymbol{y}$ of joint angles. To find f, a regression RF is trained using annotated training data $\{(I_1, \boldsymbol{y}_1), \ldots, (I_n, \boldsymbol{y}_n)\}$. The RF takes a feature vector computed on the area around a single pixel as input and returns an estimated arm configuration $\hat{\boldsymbol{y}}$. It is applied on each pixel and the results are combined as will be described in Sec. III.

A. Random Regression Forests

RFs are ensembles of decision trees, which are trained independently of each other on overlapping subsets of the whole training set. In the following, we give a brief description of RFs that will provide the basis to understand the modifications that are described in Sec. III-B. For a more in-depth review, we refer to [10].

For training one decision tree, the training samples S are hierarchically split at each node into two distinct subsets S_l, S_r, such that the samples ending up in the same leaf node are as similar as possible in terms of the target value. To achieve this, at each node a set $X = (\chi_1, \cdots, \chi_s)$ of randomly chosen split candidates is evaluated to find the candidate χ^* that maximizes the objective function:

$$\chi^* = \arg\max_{\chi \in X} \left(H(S) - \sum_{i=l,r} \frac{|S_i^\chi|}{|S|} H(S_i^\chi) \right) \tag{1}$$

where S_l^χ, S_r^χ are the left and right subset when splitting S with χ and $H(S)$ is an *impurity* function that is smaller, the more similar the elements of S are. The concrete definition of $H(S)$ depends on the split criterion that is used. The different criteria used in this paper are defined in Sec. III-B.

As output value of each leaf node, we use the mean of all associated training samples. Intuitively, this arm pose prediction shall be considered less confident, the higher the variance in the arm configurations as represented by the sample set. Therefore, we also compute a confidence value c for each leaf based on their impurity $H(S_{\text{leaf}})$. We use a *radial basis function* (RBF) kernel on the scaled impurity:

$$c = \exp\left(-\frac{\left(\nu^{-1} H(S_{\text{leaf}})\right)^2}{2 \cdot \sigma^2} \right) \tag{2}$$

The scaling factor ν is set to the maximum leaf impurity to normalize the impurities to a fixed range over the whole forest. The bandwidth σ controls the range of impurity values that result in high confidences. For our experiments we set $\sigma = 1$. However, we found that the performance of the method is not sensitive to the choice of this parameter.

At test time, the given data point computed around a pixel in the depth image traverses each tree in the forest, following the split rules learned at training time. Each tree returns the value and confidence stored at the final leaf. The pose estimate of the forest for this data point is computed as the confidence-weighted mean of the estimates of the single trees. As a confidence measure of the forest prediction for this particular pixel, the average confidence over the single trees is used.

B. DISP Distance

For training of the RF, we need a metric to measure the distance between two training samples. A common choice is the Euclidean distance, which is simple and fast to compute. Using the Euclidean distance on the angular joint configurations of a robot arm can have undesired effects, as is illustrated in Fig. 2. Therefore, in addition to the Euclidean distance we implemented the model-dependent DISP distance [2]. It is defined as follows:

$$\text{DISP}_M(\boldsymbol{y}_1, \boldsymbol{y}_2) = \max_{\boldsymbol{p} \in M} \|\boldsymbol{p}(\boldsymbol{y}_1) - \boldsymbol{p}(\boldsymbol{y}_2)\|_2 \tag{3}$$

where M is the given robot model, $\boldsymbol{p} \in M$ is a point on this model, $\boldsymbol{y}_1, \boldsymbol{y}_2$ are the two configuration samples and

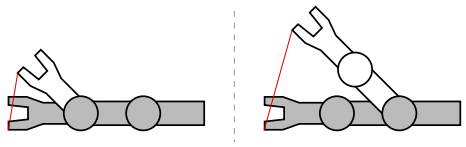


Fig. 2. Euclidean vs. DISP distance. In both examples only one joint is moved by 45°. The Euclidean distance on the angles is the same in both cases. However, the actual displacement of the end-effector varies a lot. The DISP distance, visualized by the red line, reflects this much better.

$\boldsymbol{p}(\boldsymbol{y}_i)$ denotes the position of point $\boldsymbol{p}$ when the model is in configuration $\boldsymbol{y}_i$. In Fig. 2, the DISP distance is visualized by the red line. For faster computation of the DISP distance, we use the C-DIST algorithm as proposed in [2].

III. Method

A. Depth Features

We use the same type of depth features as in [3]. They are a simplified version of the ones used in [15] and are similar to the well known BRIEF features [24]. Our features are defined as follows:

$$\varphi_i(I, \boldsymbol{p}) = I(\boldsymbol{p} + \boldsymbol{\delta}_u) - I(\boldsymbol{p} + \boldsymbol{\delta}_v) \tag{4}$$

where $I(\boldsymbol{p})$ denotes the depth value of pixel $\boldsymbol{p}$ in image I. If $\boldsymbol{p}$ has an invalid measurement or is lying outside of the image, $I(\boldsymbol{p})$ is set to a high, constant value d_{invalid}. The parameters $\boldsymbol{\delta}_u, \boldsymbol{\delta}_v$ are offsets that are fixed for a specific feature φ_i but differ between features.

We generate n_f such features by drawing the offsets $\boldsymbol{\delta}_u, \boldsymbol{\delta}_v$ randomly within a window of fixed size. By applying the features on a pixel $\boldsymbol{p}$ and stacking the values together, we get the feature vector for this pixel.

B. Training Objectives

For the training of the random forests, we compare four different training objective functions:

- Mean Squared Error (MSE)
- Mean Squared Pairwise DISP (MSPD)
- Weighted Spectral Clustering-based Split (WSC)
- Mapped Point Mean Squared Error (MPMSE)

The MSE criterion is the standard criterion for training a regression RF and is based on the Euclidean distance in the configuration space. We compare this metric, that is agnostic to the geometry of the particular problem we are addressing, to other criteria which are based on the model-based DISP distance. Our hypothesis is that these specialized criteria lead to better pose estimators for our specific task. Indeed, the MSPD criterion achieves better estimation accuracy in our evaluation, especially on unsegmented images (see Sec. IV-D). However, it has the drawback of an increased training time. The other DISP-based criteria are attempts to achieve similar accuracy while being more efficient.

The only difference between the criteria is the definition of the impurity $H(S)$ used in (1), which in the following is defined for each criterion. We define $S = \{\boldsymbol{y}_1, \dots, \boldsymbol{y}_n\}$ as the set of target values of the training samples.

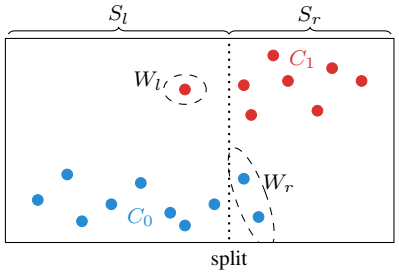


Fig. 3. Example for the WSC criterion. The colours show the result of the clustering. Assuming that the blue cluster goes left and the red one right of the split indicated by the dotted line, the one red sample on the left side forms the set W_l and the two blue samples on the right the set W_r.

1) Mean Squared Error (MSE): Minimizes the MSE by defining the impurity as the variance of the target values $\boldsymbol{y}$ of the samples. For k-dimensional target values with $k > 1$, the average variance over the single dimensions is used.

$$H_{\text{MSE}}(S) = \frac{1}{k} \sum_{i=1}^{k} \left(\frac{1}{|S|} \sum_{\boldsymbol{y} \in S} \boldsymbol{y}_i^2 - \bar{\boldsymbol{y}}_i^2 \right) \tag{5}$$

where $\bar{\boldsymbol{y}} = \frac{1}{|S|} \sum_{\boldsymbol{y} \in S} \boldsymbol{y}$ is the mean of the samples.

2) Mean Squared Pairwise DISP (MSPD): The impurity of the MSPD criterion is defined as the average distance between all pairs of samples in the set S:

$$H_{\text{MSPD}}(S) = \frac{1}{|S|^2 - |S|} \sum_{\boldsymbol{y}_1 \in S} \sum_{\boldsymbol{y}_2 \in S} \text{DISP}(\boldsymbol{y}_1, \boldsymbol{y}_2)^2 \ . \tag{6}$$

Computing $H_{\text{MSPD}}(S)$ is very time consuming due to the nested sum. We therefore developed the following objectives which also use DISP while being much cheaper to compute.

3) Weighted Spectral Clustering-based Split (WSC): At each node first a spectral clustering [25] of the samples S into two clusters C_0, C_1 is done, based on the pairwise DISP distances. We assume these clusters to represent the optimal split of the data. Therefore we evaluate the feature based split candidates by comparing the resulting splits to the clustering. The scoring is higher the better they match.

This criterion is visualized in Fig. 3. Assume that the samples of C_0 are supposed to be in the left subset S_l and the samples of C_1 in the right subset S_r. We determine the sets of "wrong samples" W_l, W_r as follows:

$$\begin{aligned} W_l &= S_l \cap C_1 \\ W_r &= S_r \cap C_0 \end{aligned} \tag{7}$$

The impurity of the split subsets is defined as the summed distance of the wrong samples in the split to the most central sample c_i of the correct cluster:

$$\begin{aligned} H_{\text{WSC}}(S_l) &= \sum_{s \in W_l} \text{DISP}(s, c_0) \\ H_{\text{WSC}}(S_r) &= \sum_{s \in W_r} \text{DISP}(s, c_1) \end{aligned} \tag{8}$$

The central sample is defined as follows:

$$c_i = \arg\min_{s \in C_i} \sum_{s' \in C_i} \text{DISP}(s, s') \tag{9}$$

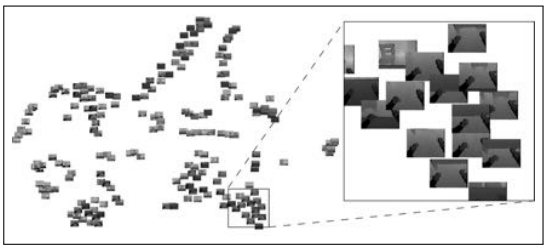


Fig. 4. Visualization of the arm poses mapped into 2-dimensional space using tSNE. Poses are visualized by the corresponding depth images.

We use this definition of the central sample instead of just computing the mean for two reasons: We avoid problems due to the circularity of the angles values ($0° = 360°$) and with our definition c_i is always in the set of original training samples which allows the use of a lookup table for the computationally expensive DISP distance.

A simpler method would be to count the "wrong samples" instead of weighting them with their distance to the cluster (i.e. $H(S_i) = \|W_i\|$). This is also less accurate, however, as it does not capture how "wrong" a sample in the split is.

Since we do not know in advance which of the two clusters better goes left and which right, we evaluate both variants and use the one with smaller sum $H_{\mathrm{WSC}}(S_l) + H_{\mathrm{WSC}}(S_r)$.

The resulting leaf impurities are not suited to compute confidence values. Therefore leaf confidences are computed as a post-processing step based on the impurity function (6) of the MSPD criterion.

4) Mapped Point Mean Squared Error (MPMSE): This impurity measure minimizes the MSE on a mapping $\boldsymbol{y}' = m(\boldsymbol{y})$ of the target values.

$$H_{\mathrm{MPMSE}}(S) = \frac{1}{k'} \sum_{i=1}^{k'} \left(\frac{1}{|S|} \sum_{\boldsymbol{y}' \in S} \boldsymbol{y}_i'^2 - \bar{\boldsymbol{y}}_i'^2 \right). \qquad (10)$$

Note that this is the same as (5) but $\boldsymbol{y}$ replaced with $\boldsymbol{y}'$ and k with k'. The mapping function m is chosen such that data points which are close according to the DISP metric are also close when measured with the Euclidean distance in the mapped space. Two mapping methods were tested: tSNE [26] and Kernel PCA (KPCA) [27].

Aside from the dimensionality of the target space, tSNE takes as input the *perplexity* p and a distance matrix $\boldsymbol{D}$ of the samples. In our case, $\boldsymbol{D}$ contains the pair-wise DISP distance between all sample points, i.e. $D_{i,j} = \mathrm{DISP}(\boldsymbol{y}_i, \boldsymbol{y}_j)$. For KPCA a Gram matrix $\boldsymbol{G}$ is used as kernel which is computed from the distance matrix as in [28]:

$$G_{i,j} = \exp\left(-\frac{D_{i,j}^2}{\sigma} \right) \qquad (11)$$

where σ is a free parameter. Fig. 4 visualizes an exemplary mapping of tSNE into a 2-dimensional space.

C. Combination of Single Pixel Predictions

The forest performs an independent pixel-wise regression to joint angles. This results in a full arm pose estimate $\hat{\boldsymbol{y}}_{\boldsymbol{p}}$ and a confidence $c_{\boldsymbol{p}}$ for each valid pixel $\boldsymbol{p} \in I$ (invalid pixels, e.g. at occlusion boundaries, are skipped so for ease of notation we assume in the following that I contains only the valid pixels). Of these "pixel-estimates", the ones of highly-confident pixels $I' = \{\boldsymbol{p} \in I \,|\, c_{\boldsymbol{p}} \geq t\}$ are chosen, where

$$t = \min_{\boldsymbol{p} \in I}(c_{\boldsymbol{p}}) + \left(\max_{\boldsymbol{p} \in I}(c_{\boldsymbol{p}}) - \min_{\boldsymbol{p} \in I}(c_{\boldsymbol{p}}) \right) \cdot t_r \qquad (12)$$

is a dynamic threshold that depends on the range of confidence values in the image. The parameter $t_r \in [0, 1]$ influences how many pixels are considered. This threshold automatically adapts to the confidence range of the given image and ensures that I' is never empty. We have observed that it usually chooses pixels of the foreground, which yield comparably good estimates (see Fig. 6).

The final pose estimate for the given image is computed as the confidence-weighted mean over the selected pixels I':

$$\hat{\boldsymbol{y}} = \frac{1}{\sum_{\boldsymbol{p} \in I'} c_{\boldsymbol{p}}} \cdot \sum_{\boldsymbol{p} \in I'} c_{\boldsymbol{p}} \cdot \hat{\boldsymbol{y}}_{\boldsymbol{p}} \ . \qquad (13)$$

D. Prior Image Segmentation

Pixels showing parts of the robot are much more informative for the estimator than pixels in the background. Further, if we want to estimate the right arm's pose, pixels showing the left arm do not provide useful information—quite the contrary they may confuse the estimator and cause bad estimates. Therefore, we expect a prior segmentation of the image into left and right arm and into background to improve the accuracy of the pose estimation.

For this segmentation, we train a classification RF on synthetic depth images. We use the same type of depth features as for the pose estimation but additionally add the pixel coordinates (x, y) and the actual depth value of the pixel. Target classes of the classification are *right arm*, *left arm* and *background*. This is similar to our previous work where additionally to the robot part labels, we also added a background class.

The resulting *segmentation forest* (SF) is applied prior to the *pose estimation forest* (PEF) described above. All pixels classified to something other than the estimated arm are set to invalid measurements which effectively removes all but the arm of interest from the image. The PEF is then applied on this segmented image, making predictions only for the remaining pixels. Note that features for the PEF are computed *after* removing the background. The whole process is illustrated in Fig. 5.

While this adds additional effort at test time for first segmenting the image, it accelerates the pose prediction in the second step significantly since no prediction has to be done for the removed background pixels.

Another way to incorporate the segmentation, that may be investigated further in future work, is not to remove background pixels but instead compute additional features on the segmentation map which are appended to the depth features computed on the original image (similar to the approach of [29]).

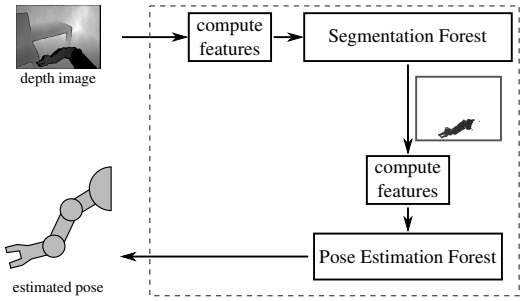


Fig. 5. First the depth image is processed by the segmentation forest to remove background pixels. The modified image is then passed to the pose estimation forest which outputs estimated joint angles for the arm.

IV. Experimental Results

We performed several experiments to evaluate the performance of our method. First we describe how we generated our dataset based on synthetic depth images. Then we use this dataset to evaluate the influence of the confidence threshold, the performance of the prior segmentation and to compare the different training objectives with each other as well as the new method to our previous work [3]. Finally we show how an estimator trained on synthetic images performs on real sensor data.

A. Training Data

We use the scikit-learn library [30] for the implementation of the RF. We trained the RF on the same synthetic depth images that we already used in [3]. To make the images as realistic as possible, we used a reimplementation of the sensor model proposed in [31], which simulates typical effects that appear in images of real RGB-D cameras like the Kinect or Xtion (e.g. depth shadows or disparity quantization). For our experiments, we used the ARM robot with two 7 DoF Barrett WAM arms and two 4 DoF Barrett hands. Depth images showing the arms are provided by a head-mounted Xtion [3]. For rendering the synthetic images we used an accurate 3D model of the robot. This makes the generated dataset specific for this particular robot but datasets for different robots can simply be created by replacing the model. The background of the images is randomized by adding some furniture (chairs, a table, closet and shelf) at random positions.

We generate $n_f = 500$ different features as defined in (4) by drawing offsets within a window of size 200×200. From each image I, we randomly draw a set of 2000 foreground pixels (showing the robot) and 1000 background pixels. On each of these pixels we apply the generated features and stack the values to form a feature vector. We set the value for invalid pixels to $d_{\text{invalid}} = 5\,\text{m}$. These settings are the same as in our previous work [3]. The target values $\boldsymbol{y}$ of the training samples are the angular joint configurations of the right arm of the robot. The arm consists of 15 joints (including the hand) so $\boldsymbol{y}$ is 15-dimensional.

When prior segmentation is used, we first train the SF and use it to segment the training images of the PEF, before features are extracted. In this case we are only sampling pixels classified as foreground and not a combination of foreground and background pixels as we do it in the case without segmentation. The training set of the SF itself consists of 3737 images and is distinct from the one of PEF to avoid that the SF is applied on its own training data when segmenting the training images of the PEF. In contrast to the PEF we reduce the number of depth features to 200 and the window size to 100×100 to accelerate feature computation. Further we add the pixel coordinates and the actual depth value of the pixel to the features. Again 3000 pixels are randomly sampled from the images but using a foreground/background ratio of 1:1 (in contrast to the PEF, where a ratio of 2:1 is used).

TABLE I
Test results of the segmentation forest.

Class	Precision	Recall	f1-Score	# pixels in test set
background	1.00	1.00	1.00	115 694 402
left arm	0.88	0.89	0.88	3 112 220
right arm	0.81	0.95	0.87	2 648 648

B. Sensitivity to Parameter t_r

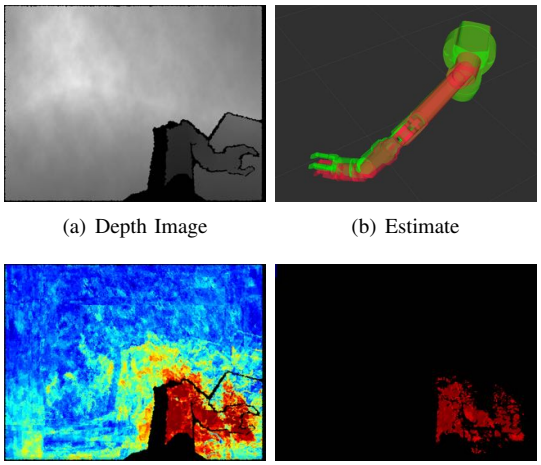
The relative confidence threshold t_r in (12) controls how many pixels of the image are used for the pose estimation. To analyse the influence of t_r on the estimation accuracy, we evaluated an RF for each training objective on unsegmented images with different values of t_r. The result is plotted in Fig. 7. In all cases, the test error decreases more or less monotonically for increasing t_r until some point near to 1. Based on Fig. 7 we set $t_r = 0.9$ for the following experiments which is a point where all training criteria perform well before for some the error increases again.

We also analysed how the threshold t varies between different images. The bounds of the confidence range (and thus also t itself) are relatively constant. The mean confidence of the image has more variation and tends to be higher for images where more parts of the robot arm are visible. Plots of the results are omitted here due to space restrictions.

Fig. 6 shows an exemplary pose estimation for one depth image. In this example no prior segmentation is used. However, prediction confidence is highest in the area of the arm. The algorithm therefore automatically selects mostly pixels on or near the arm as those are the ones that predict an arm pose with a confidence above the threshold.

C. Segmentation Forest

We use a classification RF to segment robot arms in depth images prior to pose estimation. In this section we evaluate the performance of this segmentation for the three classes *left arm*, *right arm* and *background*. Fig. 8 shows an exemplary image and the resulting segmentation map. Table I shows the test result of the whole test set. While the background is reliably detected, left and right arm are sometimes mixed up. Nonetheless, the segmentation significantly improves the pose estimation as is shown in the following. Note also that no extensive parameter search for the SF was done and pixels are classified independently of each other.



(a) Depth Image (b) Estimate

(c) Confidence (d) Used Pixels

Fig. 6. Pose estimation on a depth image using a forest trained on 160 images with the MSPD criterion. No prior segmentation is done. (a) shows the depth image, black pixels denote invalid measurements. In (b) the ground truth pose is visualized in green and the prediction of the forest in red. (c) Shows the confidence of the single pixel predictions (red means high confidence) and (d) shows them after applying the threshold with $t_r = 0.9$.

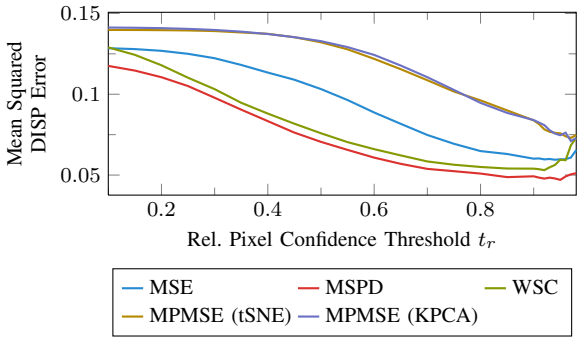


Fig. 7. Test error for different values of t_r (see III-C).

D. Compare Training Objectives

To compare the different training objectives described in Sec. III-B, we cross validated each of them on a dataset which consisted of 600 000 samples based on 200 images showing different arm poses. This was done twice: Once with only a PEF that is applied on unmodified images and once with the additional SF. The result is shown in Fig. 9.

Without segmentation, the MSPD criterion performs significantly better on this small training set than MSE at the expense of a massive increase in training time. The criterion has to be evaluated at each node in the forest for all split candidates, and is therefore slow to evaluate despite using a lookup table with precomputed DISP distances between all training samples. Table II lists exemplary training times for the different criteria on the same training set. The remaining criteria therefore attempt to take advantage of the better distance measure provided by DISP without suffering from such an extreme increase of training complexity as is the case for MSPD.

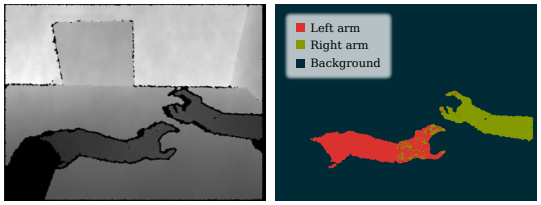


Fig. 8. Depth image and resulting segmentation map.

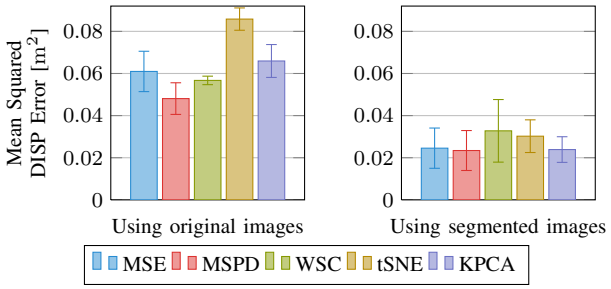


Fig. 9. Average test error and standard deviation of a 5-fold cross validation for the different training objectives. Target of the estimation is the right arm. For the left plot the unmodified depth images where used, for the right plot background and left arm were removed using a segmentation forest.

The WSC criterion indeed gains slightly better results than MSE for the case without segmentation and is much faster at training than MSPD. However, it does not achieve the same performance as the latter in terms of accuracy. A problem here could be that the "perfect splits" provided by the spectral clustering (which is based on the target values) may not be easily reproducible be the decision tree splits (which are based on feature values).

The MPMSE criterion was evaluated with two different methods to compute the point mapping: tSNE and KPCA (see Sec. III-B.4). For tSNE we set the dimensionality of the mapped points to $d = 2$ and the perplexity to $p = 40$. The parameters were determined via grid search, changing them (especially d) did not have a significant impact on the pose estimation, though. For KPCA we set $d = 10$ and $\sigma = 10$.

Fig. 10 shows an evaluation of the quality of the mappings by tSNE and KPCA using the $Q_{\text{NX}}(K)$ measure as proposed in [32]. While tSNE better preserves near neighbourhood for $d = 2$, it hardly improves when increasing d (this affirms the statement of the author of tSNE that it is not well suited for mappings into higher-dimensional spaces[1]). KPCA on the other hand performs clearly better in higher dimensions.

The results of the pose estimation go in line with this observation: RFs trained with KPCA yield considerably better estimates than RFs trained with tSNE. Without segmentation both cannot beat the traditional MSE, though. With segmented images, KPCA achieves equally good results.

E. Comparison to Previous Approach

In this section, we compare our new method to the previous one of [3]. We do this by training the classification RF of [3] on the same training set as used in Sec. IV-D. The

[1]See FAQ on `http://lvdmaaten.github.io/tsne/`

TABLE II

EXEMPLARY TRAINING TIME ON A SET OF 480 000 SAMPLES.

Training objective	Training time
MSE	7 min
MSPD	60 h
WSC*	27 min
MPMSE	7 min

*Implementation only partly parallelized.

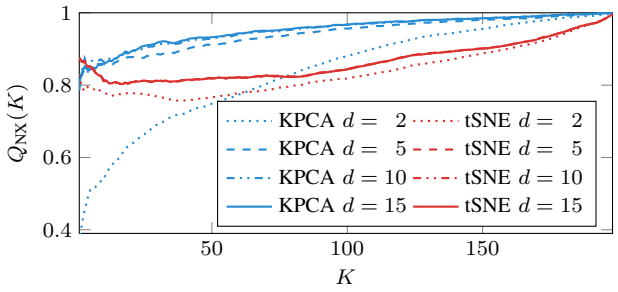
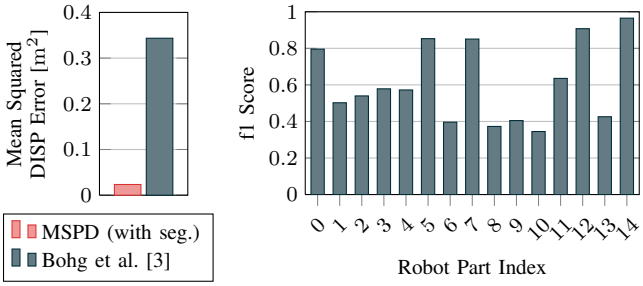


Fig. 10. Quality of the pose mapping using tSNE and KPCA with target spaces of different dimensionality. The higher a value for a specific K, the better are the K-nearest neighbours preserved on average.



(a) Comparison to [3]

(b) Part classification score of [3]

Fig. 11. Plot (a) compares the pose estimation error of the best working method of this paper (MSPD with pre-segmented images) to the previous approach of [3]. Plot (b) shows the pixel classification accuracy for the different robot parts using the method of [3] on our test set (robot part indices are the same as in [3, Table I]).

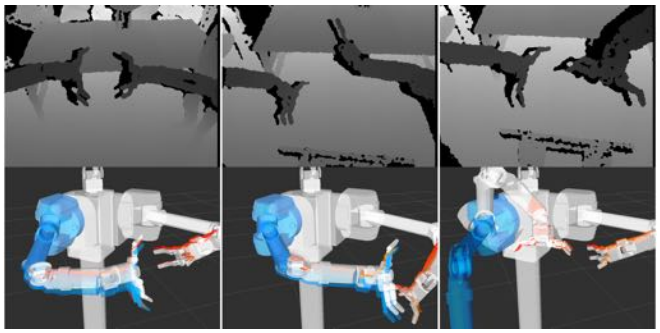


Fig. 12. Estimations on real depth images. The upper row shows the depth images, the lower one visualizes in grey the robot pose estimated from encoder readings, in orange the point cloud obtained from the depth images and in blue the estimate of our method for the right arm.

average test error of both methods is shown in Fig. 11(a). The new method considerably outperforms the previous one. Note however, that the used dataset is much smaller than the set originally used in [3] and forest and retrofitting parameters were not retuned for this. This does not affect the performance of the pixel-wise part classification. Fig. 11(b) shows the f1 scores of the classification of different robot parts on our test set. The result is similar to the one on the bigger dataset used in [3]. However, the performance of joint position estimation drops significantly which is most likely due to the considerably lower number of training examples that form the basis for voting for the relative 3D joint position.

Furthermore, none of the two methods handles the case where the arm is not visible in the image. The method proposed in [3] is most affected by this, as the joint angles are estimated by aligning the arm with the detected joint positions through inverse kinematics. If there are false positives, this will throw off the estimate completely. Indeed, the test images that caused the biggest errors were mostly those where nothing or only very little of the arm is visible.

Taking these aspects together explains the large performance difference in Fig. 11(a) and emphasizes that it is beneficial to optimize directly on error to the target values as proposed in this paper.

F. Test on Real Data

So far all evaluations were done using synthetic images. In this section we show how the RF trained on synthetic images performs when applied on real images. For doing this we used real depth images of three different static arm poses recorded with the head-mounted Xtion of our ARM robot (see Sec. IV-A). Unfortunately we have no ground truth information for the real images so we cannot give a quantitative evaluation of the estimates. However Fig. 12 qualitatively visualizes the results. To achieve these results on real data, much more training data is necessary than used above. The used RF was trained on a set of 2000 images, using the MSE training objective due to its faster training and good accuracy.

Pose estimates based on motor encoders of our robot are quite inaccurate. This can be observed by the gap between the real position of the hand as perceived by the camera (orange point cloud) and the estimated pose based on the encoders (grey model). In the first two examples the estimates of our method (blue model) look better than the ones based on the encoders. In the third one it is far off from the real arm pose. This can be explained by the pose which is far away from the poses contained in the training set. This can be mitigated by using a larger training set that better covers the configuration space of the arm.

V. CONCLUSION AND FUTURE WORK

Based on synthetically generated depth images, we trained a regression RF to estimate robot arm poses in single depth images and based on simple depth features. We showed that this considerably improved our previous approach [3] in terms of accuracy. At the same time, we reduced the computational effort at test time as the new method trains

an RF to directly regress to joint angles. In contract, the former approach did a detour by first classifying the pixel's robot part and then estimating the 3D position of each joint through time-consuming clustering of the single pixel votes.

The proposed method works on a frame-by-frame base without the need for initialization or knowledge about previous frames. Furthermore, it can handle raw depth images without the need of any preprocessing as pixel selection based on confidence automatically chooses pixels mostly on or near to the arm. However, we showed that the accuracy can be improved using a separately trained segmentation forest for prior segmentation of the arm of interest.

We showed how different training objectives that use DISP rather than Euclidean distance influence the accuracy of the pose estimation. We expected the DISP distance to work better, as it better reflects the displacement of the arm. On unsegmented images the DISP-based MSPD objective achieves indeed higher accuracy than the traditional Euclidean distance-based MSE objective at the cost of a computationally more expensive training. However, with prior segmentation the difference decreases—MSE achieves almost the same results here while being much more efficient. We evaluated other DISP-based objectives that are computationally less expensive than MSPD but none of them could could considerably beat MSE in terms of accuracy.

So far, no explicit analysis of the robustness against occlusions was done. This would be interesting as parts of the arm can be hidden by the other arm or due to self-occlusion. Further it would be interesting to repeat the experiments with another robot that has a different shape and to compare different types of depth features. The presented method assumes a fixed camera pose. As the robot can move its head, it would be beneficial to improve the method to handle a moving camera (note that our previous work [3] can handle this, using the encoder readings of the head joints). As already mentioned above, an alternative to using the SF to remove background, would be to generate a probabilistic segmentation map and compute additional features based on it. The PEF would then be trained on features coming partly from the depth image and partly from the segmentation map. This idea was already successfully used in [29] and first experiments on our task showed significant improvement compared to when no segmentation is used.

REFERENCES

[1] P. Pastor, *et al.*, "Learning task error models for manipulation," in *Proc. IEEE Int. Conf. Robotics and Automation*, May 2013, pp. 2612–2618.

[2] L. Zhang, *et al.*, "C-DIST: Efficient Distance Computation for Rigid and Articulated Models in Configuration Space," in *Proc. ACM Symp. Solid and Physical Modeling*, 2007, pp. 159–169.

[3] J. Bohg, *et al.*, "Robot Arm Pose Estimation through Pixel-Wise Part Classification," in *Proc. IEEE Int. Conf. Robotics and Automation*, 2014.

[4] X. Gratal, *et al.*, "Scene Representation and Object Grasping Using Active Vision," in *IROS'10 Workshop on Defining and Solving Realistic Perception Problems in Personal Robotics*, 2010.

[5] N. Vahrenkamp, *et al.*, "Visual Servoing for Humanoid Grasping and Manipulation Tasks," in *8th IEEE-RAS Int. Conf. Humanoid Robots*, Dec 2008, pp. 406–412.

[6] X. Gratal, *et al.*, "Visual Servoing on Unknown Objects," *Mechatronics*, vol. 22, no. 4, pp. 423–435, 2012.

[7] P. Hebert, *et al.*, "Combined Shape, Appearance and Silhouette for Simultaneous Manipulator and Object Tracking," in *Proc. IEEE Int. Conf. Robotics and Automation*, May 2012, pp. 2405–2412.

[8] M. Krainin, *et al.*, "Manipulator and Object Tracking for In-Hand 3D Object Modeling," *Int. J. of Robotics Research*, vol. 30, no. 11, pp. 1311–1327, 2011.

[9] T. Schmidt, *et al.*, "DART: Dense Articulated Real-Time Tracking," in *Proc. Robotics: Science and Systems*, Berkeley, USA, July 2014.

[10] A. Criminisi and J. Shotton, *Decision Forests for Computer Vision and Medical Image Analysis*. Springer Science & Business Media, 2013.

[11] M. Dumont, *et al.*, "Fast Multi-Class Image Annotation with Random Windows and Multiple Output Randomized Trees," in *Proc. Int. Conf. Computer Vision Theory and Applications*, vol. 2, 2009, pp. 196–203.

[12] J. Gall and V. Lempitsky, "Class-specific hough forests for object detection," in *Decision Forests for Computer Vision and Medical Image Analysis*. Springer, 2013, pp. 143–157.

[13] M. Godec, *et al.*, "Hough-Based Tracking of Deformable Objects," in *Decision Forests for Computer Vision and Medical Image Analysis*. Springer, 2013, pp. 159–173.

[14] V. Lepetit and P. Fua, "Keypoint Recognition Using Random Forests and Random Ferns," in *Decision Forests for Computer Vision and Medical Image Analysis*, 2013, pp. 111–124.

[15] J. Shotton, *et al.*, "Efficient Human Pose Estimation from Single Depth Images," *Trans. on Pattern Analysis and Machine Intelligence*, 2012.

[16] ——, "Semantic texton forests for image categorization and segmentation," in *IEEE Conf. Computer Vision and Pattern Recognition*, 2008, pp. 1–8.

[17] O. Stern, *et al.*, "Automatic Localization of Interest Points in Zebrafish Images with Tree-based Methods," in *Proc. 6th IAPR Int. Conf. Pattern Recognition in Bioinformatics*, ser. PRIB'11, 2011, pp. 179–190.

[18] F. Michel, *et al.*, "Pose estimation of kinematic chain instances via object coordinate regression," in *Proc. British Machine Vision Conf.*, September 2015, pp. 181.1–181.11.

[19] M. Hofer and H. Pottmann, "Energy-minimizing Splines in Manifolds," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 284–293, Aug. 2004.

[20] Y. J. Kim, *et al.*, "Fast Swept Volume Approximation of Complex Polyhedral Models," in *Proc. 8th ACM Symp. Solid Modeling and Applications*, ser. SM '03, New York, NY, USA, 2003, pp. 11–22.

[21] J.-C. Latombe, *Robot Motion Planning*, ser. The Springer Int. Series in Engineering and Computer Science. Springer US, 1991.

[22] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006.

[23] P. Xavier, "Fast Swept-Volume Distance for Robust Collision Detection," in *Proc. IEEE Int. Conf. Robotics and Automation*, vol. 2, Apr 1997, pp. 1162–1169 vol.2.

[24] M. Calonder, *et al.*, "BRIEF: Binary Robust Independent Elementary Features," in *European Conf. Computer Vision*, 2010, pp. 778–792.

[25] U. von Luxburg, "A Tutorial on Spectral Clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, Dec. 2007.

[26] L. Van der Maaten and G. Hinton, "Visualizing Data using t-SNE," *J. Machine Learning Research*, vol. 9, no. 2579-2605, p. 85, 2008.

[27] B. Schölkopf, *et al.*, *Kernel Principal Component Analysis.* Cambridge, MA: MIT Press, 1999, pp. 327–352.

[28] R. Detry, *et al.*, "Generalizing grasps across partly similar objects," in *Proc. IEEE Int. Conf. Robotics and Automation*, 2012, pp. 3791–3797.

[29] Z. Tu and X. Bai, "Auto-Context and Its Application to High-Level Vision Tasks and 3D Brain Image Segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 10, pp. 1744–1757, 2010.

[30] F. Pedregosa, *et al.*, "Scikit-learn: Machine Learning in Python," *J. Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[31] M. Gschwandtner, *et al.*, "BlenSor: Blender Sensor Simulation Toolbox," in *Advances in Visual Computing*, ser. Lecture Notes in Computer Science, G. Bebis *et al.*, Eds. Springer Berlin Heidelberg, 2011, vol. 6939, pp. 199–208.

[32] J. A. Lee and M. Verleysen, "Quality Assessment of Dimensionality Reduction: Rank-based Criteria," *Neurocomputing*, vol. 72, no. 7, pp. 1431–1443, 2009.