A unified energy minimization framework for model fitting in depth *

Carl Yuheng Ren and Ian Reid

Oxford University, Dept. Engineering Science, Parks Road, Oxford, OX1 3PJ, UK
{carl,ian}@robots.ox.ac.uk

Abstract. In this paper we present a unified energy minimization framework for model fitting and pose recovery problems in depth cameras. 3D level-set embedding functions are used to represent object models implicitly and a novel 3D chamfer matching based energy function is minimized by adjusting the generic projection matrix, which could be parameterized differently according to specific applications. Our proposed energy function takes the advantage of the gradient of 3D level-set embedding function and can be efficiently solved by gradients-based optimization methods. We show various real-world applications, including real-time 3D tracking in depth, simultaneous calibration and tracking, and 3D point cloud modeling. We perform experiments on both real data and synthetic data to show the superior performance of our method for all the applications above.

1 Introduction

Since the recent release of consumer-priced video-rate depth sensors, there has been an explosion in the field of tracking[1], 3D reconstruction [2] and object recognition [3], etc. using depth data. A large proportion of these works use a model-based paradigm; i.e. given an object model, try to align a known object model with the observed depth map.

Model-based approaches generate model hypotheses and evaluate them on the available visual observations, defining an objective function that measures the discrepancy between the visual cues that are expected from a model hypothesis and the observed ones. Based on how the optimization of this objective is performed, we can define two categories of methods: sampling based solvers and gradient-based solvers.

Sampling based solvers [1, 4] rely on many evaluations of energy function at arbitrary points in the multidimensional model hypothesis space. The authors of [1] solve the energy minimization problem using a variant of Particle Swarm Optimization (PSO). The GPU implementation of the their method yields near real-time performance (15Hz). Another recent work in this category is by Ryohei Ueda [4], which formulates a very similar energy function to [1]. The energy functions in both works [1, 4] above measure the discrepancy between the expected

* This work is funded by REWIRE project (Grant N. 287713) under the 7-Framework Programme.

depth generated by pose hypothesis and the real depth. In Ryohei Ueda's work [4], a particle filter is used for for solving the optimization. Methods in this category usually require extensive computational power, because evaluation of the energy function online is usually computationally expensive.

Gradient-based solvers try to use the depth image gradient to guide the search for the best hypothesis in the high dimensional hypothesis space. Iterative Closest Point (ICP) [5] framework is the most commonly used method in this category. Temporary point correspondences between an observed point cloud and the known model are established by finding the nearest neighbor for each vertex on the object model, and the resulting sum of pair-wise square-distance based energy function is differentiated with respect to model hypothesis parameters. Using the depth image gradient in this way can yield much more efficient optimization with many fewer energy function evaluations. However, establishing point correspondences can be as computational expensive as energy function evaluation. Furthemore, the depth image gradients are often very noisy, thus they can not be used without smoothing or resolution hierarchy.

This paper presents a unified energy minimization framework for model fitting problems in depth. Instead of projecting the model down to the image plane and measuring the discrepancy between expected image cues and the observed ones, the depth images permit us to back project all observed pixels into the object coordinate, where a level-set embedding function is defined to encode the object model implicitly. Our formulation of energy function is a natural extension to the chamfer-matching based tracking method [6] in 2D. This proposed method has the following advantages over existing model based approaches:

1. The search can be solved efficiently using a gradient-based solver, but instead of relying on the noisy gradient of the depth image, our method takes advantage of the smooth gradient of level-set embedding function to guide the search. Furthermore, for a given level-set embedding, the per-voxel gradients are independent of the pose parameters, and can therefore be computed in advance for computational efficiency (for objects with constant shape).
2. Instead of "rendering" the model into each image, which would require z-buffering or other means to determine depth order, our method back projects the depth pixels into the object coordinate frame, aiming for alignment with the zero-level of the level-set embedding function. Thus no z-buffer is required and the method is inherently more suitable for the parallelization on a GPU.
3. Our framework is a region-based method, so no point correspondences are required. As we show in subsection 3.2, our framework can, for example, be used for calibrating the intrinsic parameters of a depth camera without the need to establish any point correspondences (manually or automatically).

2 Notation

Let I_d be a depth image obtained from the depth camera, and the depth image domain denoted by Ω. Without lose of generality, a pixel at (u, v) in the depth image is has homogeneous coordinates $\mathbf{x} = [ud, vd, d]^T \in \Omega$ with explicit depth

d. A depth point $\mathbf{x}$ in the projected image region of the object is projected from a 3D object surface point $\mathbf{X} = [x, y, z, 1]^T$ in the object frame as:

$$\mathbf{x} = P\mathbf{X} \tag{1}$$

where P is a 3×4 projection.

An object model is implicitly represented by a 3D level-set embedding function Ψ defined in the object coordinate frame. The surface of an object can be recovered from the zero-level of the level-set embedding function i.e. $\Psi(\mathbf{X}) = 0$. The value of Ψ increases monotonically in space as a point moves further from the nearest surface of the object model. The most convenient form of Ψ is thus the 3D distance transform Φ, which we use to formulate variants of Ψ for different applications.

3 Energy Function

Given that a pixel in the projected object region of the depth image is projected from a 3D point on the object model surface, its value in object coordinates $\mathbf{X}$ can be obtained exactly via back-projection from $\mathbf{x}$.

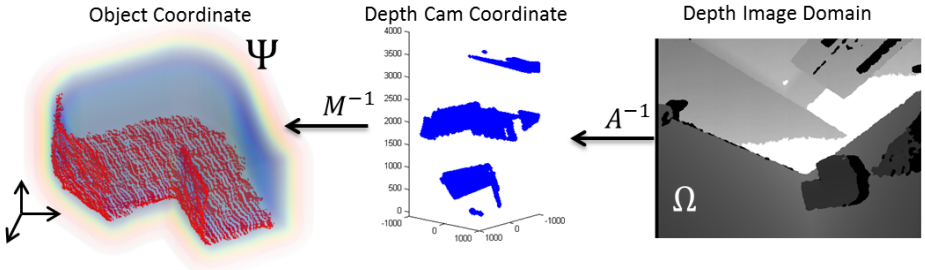

Fig. 1. Example of our back-projection formulation, from left to right: all pixels on the depth image are first unprojected into depth camera coordinate by intrinsic matrix A, then inverse transformed by extrinsic matrix M into the object coordinate, where the level-set embedding function is defined.

If the projection matrix P is correctly recovered, all depth pixels $\mathbf{x}$ in the object region should be back-projected onto the model surface $\Psi(\mathbf{X}) = 0$. Fig. 1 showcases an example of formulating such projection by the simplest pinhole model, which decomposes P into an intrinsic (calibartion) matrix A and extrinsic (pose) matrix M. Our base energy function is defined as:

$$E = \sum_{\mathbf{x} \in \Omega} \Psi(\mathbf{X}) \tag{2}$$

Thus an evaluation of the energy function comprises back projecting all the pixels in the depth image into the local coordinate of the object and summing the corresponding values in the level-set embedding function. As discussed above, a convenient form of embedding function is a distance transform, though more specifically we use a truncated distance transform since points that back-project

outside the local volume of the object are assigned a nominal "large" distance value. In this way, the pixels in the object region are automatically segmented from the background and the whole energy function will take minimum value when the model is aligned with the depth point cloud.

3.1 Tracking rigid 3D objects

The first and simplest application of this energy function is to track rigid 3D objects in depth. Assuming known object model (encoded in 3D distance transform Φ), we formulate the level-set embedding function Ψ using German-McClure object function, which is robust to outlier pixels in the object region:

$$\Psi = \frac{\Phi^2}{\Phi^2 + \sigma} \tag{3}$$

where σ is a constant parameter to determine the width of the energy function. The projection matrix P can be decomposed into intrinsic matrix A and 3×4 extrinsic matrix M: $P = AM$. Assuming calibrated camera (i.e. A known), P is parameterized only by the pose of the object p, we differentiate the energy function w.r.t. p

$$\frac{\partial E}{\partial p} = \sum_{\mathbf{x} \in \Omega} \frac{2\sigma\Phi}{(\sigma + \Phi^2)^2} \frac{\partial \Phi}{\partial \mathbf{X}} \frac{\partial \mathbf{X}}{\partial p} \tag{4}$$

Where $\frac{2\sigma\Phi}{(\sigma+\Phi^2)^2} \frac{\partial \Phi}{\partial \mathbf{X}}$ is the gradient of the level-set embedding function, which could be computed in advance. We use Levenberg-Marquardt method to compute the step at each iteration then use local frame to update the pose:

$$\tilde{p} = -\left(J_E^T J_E + \alpha \mathbf{diag}\left[J_E^T J_E\right]\right)^{-1} \frac{\partial E}{\partial p} \tag{5}$$

$$M_n \leftarrow \tilde{M}(\tilde{p}) M_{n-1} \tag{6}$$

where J_E is the Jacobian matrix of the energy function. The automatic step-size controlling parameter α decreases when a step causes the energy function to decrease and increases when energy function increases.

3.2 Simultaneous tracking and calibration

When the intrinsic matrix A and the extrinsic matrix M are both unknown, the projection matrix P is parameterized by both the intrinsic parameters $k = [fx, fy, cx, cy, kc]$ and pose p. We formulate the level-set embedding function using square energy function, and then we differentiate the energy function w.r.t the joint intrinsic-pose parameter $\lambda = [k, p]$:

$$\Psi = \Phi^2, \quad \frac{\partial E}{\partial \lambda} = \sum_{\mathbf{x} \in \Omega} 2\Phi \frac{\partial \Phi}{\partial \mathbf{X}} \frac{\partial \mathbf{X}}{\partial \lambda} \tag{7}$$

Note that the reason that we use the square energy function instead of German-McClure robust object function for simultaneous calibration and tracking is that

square energy function provides better convergence around the true value of intrinsics and fewer local minima when optimizing in the joint intrinsic-pose parameter space. Also, we assume that when users try to calibrate a camera using our method, the frames that are captured have few outliers in the object region. Again, we use Levenberg-Marquardt method to obtain the optimal intrinsic parameter k and pose p at each frame and form an estimate via weighted least squares using all values up to frame t:

$$\hat{k} = arg_k \max \left\{ \sum_{i=1}^{t} (k - k_i)^T \Sigma_i^{-1} (k - k_i) \right\} \tag{8}$$

where Σ_i is the covariance of intrinsic parameter k at frame i, and is approximated by the inverse of the Hessian.

3.3 Point cloud modeling with adaptive primitive models

In this application, we use our energy function to model a point cloud with adaptive primitive object models (spheres, cylinders and cones, etc.) with unknown size. Given the intrinsic parameters and each object class, our energy function can fit shape primitives into a point cloud with correct scale. Assuming the scale of object model is along x, y, z axis in object frame, P is decomposed:

$$P = AMS, \quad S = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{9}$$

where S is the scaling matrix. Now the 3D point $\mathbf{X}$ is a function of both the 3D pose p and the scaling parameters $[s_x, s_y, s_z]$, We optimize the energy function w.r.t. the joint parameter $\lambda = [s_x, s_y, s_z, p]$. We follow the same optimization method in 3.2 to jointly recover the pose of the object and its size. For multiple frames, we use weighted least squares as in Section 3.2 to estimate the scale.

4 Experiment and evaluation

We tested our energy function on the three applications we introduced above. The performance of our energy function is evaluated both quantitatively and qualitatively for the application of tracking rigid 3D object and stimulations calibration and tracking. We also showcase the real world result of our energy function for the application of point cloud modeling. The initial poses of the objects are manually initialized in the following experiments, but motion detection and object recognition algorithms can also be used for automatic initialization.

4.1 Tracking rigid 3D object

We have tested our energy function for tracking rigid 3D object on various real-world sequences, Fig 2 shows an example of using our energy function to track an real object with heavy occlusion. As is shown, even when more than half of the

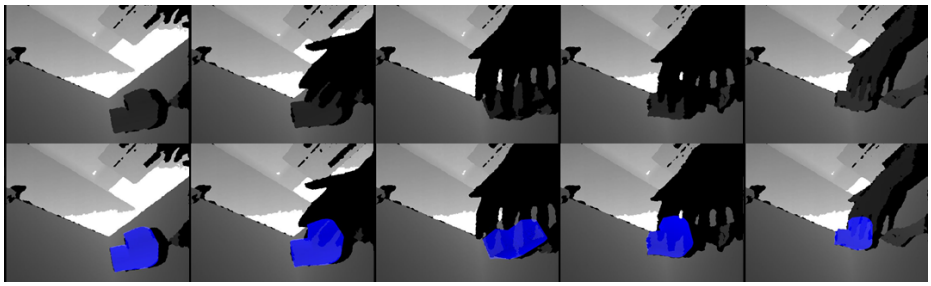

Fig. 2. Film strips showing our algorithm successfully tracking a real rigid object through heavy occlusion. The upper row shows the observed depth sequence, while we show the tracking result on the lower row.

object is occluded by hand, the tracker still did not lose track. This is because our method back projects the depth point on the image into object coordinate and use the gradient of the level-set embedding function to guide the search for the best pose. In this way, our energy function is naturally robust to occlusion and missing data: as long as the back-projected points encodes sufficient information of the location of the object, our method can converge correctly.

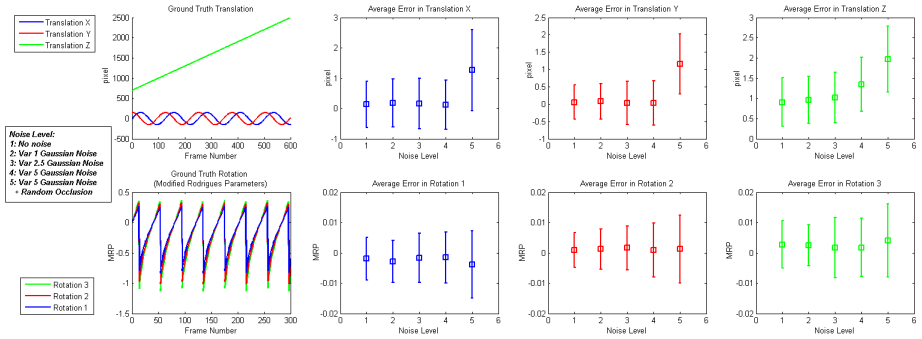

Fig. 3. Quantitative evaluation of the precision and robustness of our method for tracking 3D rigid object on synthetic data, with respect to different level of noise and occlusion. The error in translation is measured in pixel while rotation is measured in modified rodrigues parameters.

Because it is very difficult to evaluate the performance of 3D rigid object tracker without ground truth data, we use known model to generate synthetic depth sequences and run our tracker on such sequences to evaluate the precision and robustness of our energy function for tracking 3D rigid objects. As is shown in Fig. 3, we generated synthetic depth sequences with 5 levels of noise and occlusion, ranging from no noise to variance 5 gaussian noise plus random occlusion. The ground truth translation (in pixel) and rotation (in modified rodrigues parameters, MRP) for 600 frames are shown in the first chart on the first and second row.The rest of the charts shows the mean and 2-times standard deviation of the error in translation and rotation through all 600 frames on different noise level. We can see that even with the highest noise and occlusion, the tracking errors of our method can still remains very low (< 2.5 pixels in all translation and < 0.01 MRP in all rotation).

4.2 Simultaneous tracking and calibration

The second set of experiments is to test the performance of our method for simultaneous calibration and tracking. We first evaluate calibration result using synthetic data. With known intrinsic camera model, we generated depth sequences using the same method in subsection 3.1 with 3 different noise levels, then we use our energy function to run simultaneous calibration and tracking algorithm on these sequences. Fig. 4 shows the values of intrinsic parameters estimated from 300 synthetic frames. With low noise, our method can recover the intrinsic parameter accurately from less than 200 tracking frames, even with variance 5 gaussian distributed noise on the observed depth sequence, our method can still recover some reasonable intrinsic parameters.

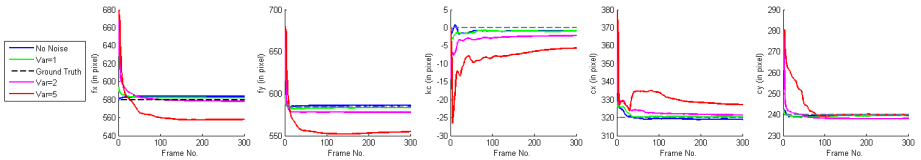

Fig. 4. Quantitative evaluation of our method for calibration on synthetic data, with respect to different level of noise, all intrinsic parameters are in pixels.

We have also tested our method on real data. Note that, in order to get precise calibration result, a simple object (such as a box) is not sufficient since it has ambiguity. A slightly more complex calibration object (see Fig. 5) is used for the experiment. We initialize the tracking with an initial guess of the intrinsics ($fx = 600, fy = 600, cx = 300, cy = 200, kx = 0$), then track the object for 500 frames to estimate the intrinsic matrix. Fig. 5 shows some example frames.

Fig. 5. The leftmost images shows our calibration object, Film strips to the right shows an example sequence that we used for simultaneous calibration and tracking.

Reprojection error is the most commonly used criteria to evaluate intrinsic calibration, however, without point correspondences in depth, we can not use this criteria. Thus we use similar criteria as in D. Herrera et al. [7], which is based on orthogonal planes, to evaluate the quality of our intrinsic estimation. As is illustrated in 6, we use the depth camera which we try to calibrate to capture a depth frame of three known orthogonal planes, then we un-project all depth pixels into the camera coordinate and compute the angle between the normals of the three planes. If the intrinsic matrix is correct, the angle between the three norms should all be 90°.

With this criteria, we compare our calibration result with the standard checkerboard calibration method with non-linear refinement [8] and the the state-of-the-art Kinect calibration toolbox by D. Herrera et al. [7]. Note that,

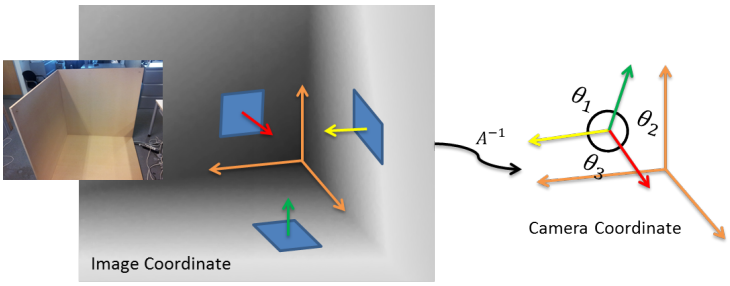

Fig. 6. Illustration of the criteria for the evaluation of intrinsic parameters: given a depth image of three orthogonal planes, we unproject all depth pixels from image coordinate to camera coordinate, then we compute the angle between the norms of the three orthogonal planes as criteria. If the intrinsic matrix is correctly recovered, the angle between the tree norm should all be 90°.

for the standard checker board method, we attached checker board patter on a piece of glass then cut off the white area, so that we can do automatic corner detection in depth image. In Table 1 we shows the experiment result, the three norm angles computed from our initial guess are also listed in the table for comparison. As is shown, our method shows the best result for recovering orthogonal plane normals. The disadvantage of our method compared to D. Herrera et al. [7] is that, our method can not calibrate the relative pose between the depth camera and the color camera. [7] requires the user to define the boundary of the calibration plane at all frames, however, given a known object model, our method can calibrate the depth sensor with one-click initialization on the first frame. No point correspondences, either by manual clicking or automatic corner detection is required.

Table 1. Quantitative evaluation of the performance of our method for calibration on real data: criteria are based on the angle between the norms of up-projected orthogonal planes, all angles are in degree.

Method	θ_1	θ_2	θ_3	Overall Err
Ours	**90.1849**	91.6065	**89.6874**	**2.1040**
Checker Board [8]	90.8706	92.1943	87.9618	5.1031
D. Herrera et al. [7]	91.6082	**89.2167**	90.3549	2.7464
Initial guess	88.4047	87.9643	94.1769	7.8079

4.3 Point cloud modeling with adaptable primitive models

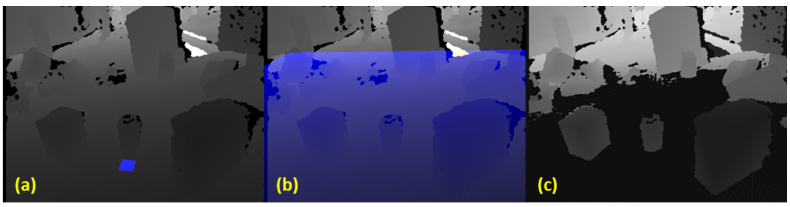

Fig. 7. Illustration of the ground plane removal step: a) fitting a plane patch to any part of the ground plane. b) Extend the plane patch. c) Ground plane is segmented out by removing all depth pixels that are close to the extended plane patch.

Due to the lack of ground truth data, we only qualitatively showcase the application of our energy function for point cloud modeling with adaptable primitive models. Fig. 7 illustrates the method we use to remove ground plane. Given the depth map of a scene, we first remove the ground plane. We do this by fitting a plane patch to any part of the ground plane using our 3D rigid object tracker. Then we extend the plane to cover whole depth image,and compute the discrepancy between the extended plane and the depth image on each pixel. All pixels that has small discrepancies are removed as ground plane.

After removing the ground plane, we initialize primitive models with arbitrary size, then run the rigid 3D object tracker to align the primitives to the point cloud. The primitive models will not fit the point cloud perfectly since the size is not correct, but the tracking result will be used as initialization for the shape adaptation step. In the final shape adaptation step we used the energy function parameterized by extrinsic and scaling matrix to simultaneously workout the size and position of the primitive models. Results are shown in Fig. 8.

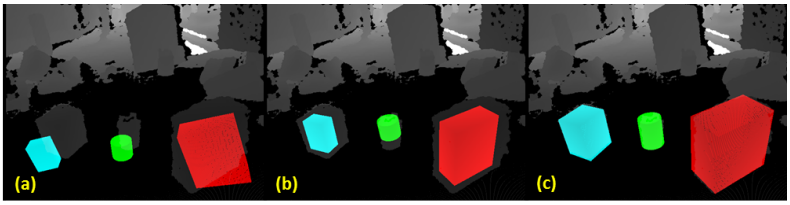

Fig. 8. Illustration of the model adaptation step: a) primitive models are initialized with arbitrary size; b) using our rigid 3D object tracking energy function, models are fitted into point cloud as rigid objects; c) shape adaptations energy function is used to simultaneously deform and track the primitives towards final fitting.

Finally, we also tested the speed of the GPU implementation of the above 3 experiments on a Core-i7 870 (2.93GHz) machine with GTX680 graphic card: rigid tracking (6 pose parameters) runs at 10ms/frame@640×480; simultaneous calibration & tracking (11 parameters) and point cloud modeling (9 parameters) runs at around 20ms∼30ms/frame@640×480. This shows that our method yields real-time performance in all above applications.

5 Conclusion and future work

In this paper we have presented a novel generic framework for model fitting problems in depth, which could be used for tracking 3D rigid objects, simultaneous calibration and tracking, point cloud modeling, shape adaptation and etc. Our framework formulates the model fitting problem as an optimization problem for the best projection matrix that back projects depth points onto the zero-level of a level-set embedding function, which encodes the 3D shape of the object. The framework is generic in that various energy functions can be formulated for different purposes. By leveraging the gradient of the level-set embedding function, the formulated object functions can be solved efficiently with gradient-based optimization methods. The per-pixel back projection mechanism is perfectly parallelizable thus real-time GPU-based applications can be

implemented using our framework. We have demonstrated the performance of our framework in various applications with extensive experiment on both real and synthetic data.

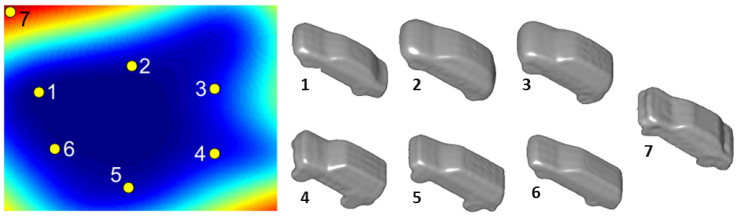

Fig. 9. Illustration of a 3D latent space of 3D car shapes: valid 3D car models (1∼6) can be generated from the high-probability (blue) area in the latent space [9].

One possible extension to the current framework is to apply it to track deformable or articulated object classes. In subsection 3.1, 3.2 and 3.3, Φ is fully defined and constant for a given object model, however, we note that Φ could also be generated from a point in latent object shape space as well. For example, the authors in [9] capture 3D shape variance in Gaussian Process Latent Variable Models (GPLVM) [10] (Fig. 9 shows an example of such latent space). Our optimization would then proceed as a minimization simultaneously over the latent space parameters governing the shape and the pose parameters, retaining all of the advantages of our method.

References

1. Iason Oikonomidis, N.K., Argyros, A.: Efficient model-based 3d tracking of hand articulations using kinect. In: Proceedings of the British Machine Vision Conference, BMVA Press (2011) 101.1–101.11
2. Newcombe, R.A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A.J., Kohli, P., Shotton, J., Hodges, S., Fitzgibbon, A.W.: Kinectfusion: Real-time dense surface mapping and tracking. In: ISMAR, IEEE (2011) 127–136
3. Koppula, H.S., Anand, A., Joachims, T., Saxena, A.: Semantic labeling of 3d point clouds for indoor scenes. In: NIPS. (2011) 244–252
4. Ueda, R.: Tracking 3d objects with point cloud library (2012) `pointclouds.org`.
5. Besl, P.J., McKay, N.D.: A method for registration of 3-d shapes. IEEE Trans. Pattern Anal. Mach. Intell. **14** (1992) 239–256
6. Thayananthan, A., Stenger, B., Torr, P.H.S., Cipolla, R.: Shape context and chamfer matching in cluttered scenes. 2012 IEEE Conference on Computer Vision and Pattern Recognition **1** (2003) 127
7. C, D.H., Kannala, J., Heikkila, J.: Joint depth and color camera calibration with distortion correction. IEEE Transactions on Pattern Analysis and Machine Intelligence **99** (2012)
8. Zhang, Z.: A flexible new technique for camera calibration. Pattern Analysis and Machine Intelligence, IEEE Transactions on **22** (2000) 1330 – 1334
9. Prisacariu, V., Reid, I.: Simultaneous monocular 2d segmentation, 3d pose recovery and 3d reconstruction. Technical report, University of Oxford (2012)
10. Lawrence, N.D.: Probabilistic non-linear principal component analysis with gaussian process latent variable models. Journal of Machine Learning Research **6** (2005) 1783–1816