# Enabling Efficient Registration using Adaptive Iterative Closest Keypoint

Johan Ekekrantz[1], Andrzej Pronobis[2], John Folkesson[1] and Patric Jensfelt[1]

***Abstract*— Registering frames of 3D sensor data is a key functionality in many robot applications, from multi-view 3D object recognition to SLAM. With the advent of cheap and widely available, so called, RGB-D sensors acquiring such data has become possible also from small robots or other mobile devices. Such robots and devices typically have limited resources and being able to perform registration in a computationally efficient manner is therefore very important. In our recent work [1] we proposed a fast and simple method for registering RGB-D data, building on the principle of the Iterative Closest Point (ICP) algorithm. This paper outlines this new method and shows how it can facilitate a significant reduction in computational cost while maintaining or even improving performance in terms of accuracy and convergence properties. As a contribution we present a method to efficiently measure the quality of a found registration.**

## I. INTRODUCTION

Data registration is the natural next step after acquisition of sensory data. The goal is to align two frames of sensor data of the same scene taken from different locations. Registration is often used as a way to replace or enhance odometry obtained from wheel encoders. Registration is important because a robot's behavior is based on its world model and that world model requires accumulation of data in a consistent reference frame. Therefore, a more accurate data registration allows the robot to make better inferences and decisions.

The recent advancements in RGB-D cameras have led to increasing use of range image data in robotics. The availability of both depth and visual information can largely simplify the registration itself. In this work, we focus on the problem of registration of RGB-D views and actively exploit the visual content to improve both accuracy and efficiency.

In [1], we present Adaptive Iterative Closest Keypoint (AICK), a registration algorithm for RGB-D views which builds on the idea of Iterative Closest Point (ICP) [2]. Algorithms based on the principle of ICP are able to provide very accurate estimations, given an initial transformation that is close to the final result. Unfortunately, the performance of standard ICP often deteriorates steeply with the decrease of the quality of the initial guess, as often happens in case of registration of views captured during fast sensor rotations. Additionally, noise can drastically affect the convergence of the iterative optimization method, with local minima being a common problem.

[1]The authors are with the Centre for Autonomous System at KTH Royal Institute of Technology, SE-100 44 Stockholm, Sweden {ekz,johnf,patric}@csc.kth.se

[2] A. Pronobis is with the Robotics and State Estimation Lab at the University of Washington. pronobis@cs.washington.edu

AICK preserves the accuracy of ICP for small transformations, while providing a drastic improvement of robustness to larger view rotations and translations without the need for an initial guess given sufficient overlap between the frames. Our algorithm exploits both depth and visual information and relies on keypoints detected in images associated with 3D positions in the local reference frame and a visual descriptor. The key property of the algorithm is the ability to weigh the importance of the visual descriptor and the 3D position while iteratively optimizing the transformation. This allows us to exploit the distinctiveness of appearance features for improved initial robustness and accuracy of point locations for the final precision.

In this paper we compare the proposed method to generalized ICP (GICP) [3], the 3D normal distribution transform (3D-NDT) [4] and a method based on RANSAC [5] and keypoints which we will call 3-point RANSAC and show how our method provides a significant reduction in computational cost without sacrificing performance and improving it significantly in most use cases. The evaluation of the four algorithms is performed on a publicly available dataset [6] and we base our quantitative analysis on an established benchmarking procedure and performance measure [6]. In this paper we also present an efficient way to assess the quality of the registration between two frames.

In the remaining parts of the paper, we first provide an overview of registration methods. Section III provides details of the proposed algorithm. Section IV covers the setup for the experimental evaluation. Finally, we present the results of the experimental evaluation in Section V.

## II. RELATED WORK

Most of the point cloud registration methods are based on the Iterative Closest Point (ICP) algorithm introduced in [2]. The most computationally expensive part of ICP is typically finding the closest points. The standard way of performing the matching is to use nearest neighbour matching in Euclidean space. This has a complexity of $\mathcal{O}(N^2)$ in a naive implementation. A common way to speed this up is to use a kd-tree (or a set of trees) which reduces the complexity to $\mathcal{O}(Nlog(N))$.

Each point cloud is a sample of the real world and even small perturbations in the sensor pose can lead to sampling different structures or parts thereof. A common way to address this is to make a parametric model and then, for example, fit the points in one frame against planes in the other as in [7] or more recently in the GICP algorithm [3] where both point clouds are models with planar surfaces.

The 3D normal distribution transform (3D-NDT) [4], [8] fit Gaussian ellipsoids to the data which both address the issue of noise and reduces the dimensionality of the data, thus speeding up the processing. Similar work has been presented in [9]. The Multi-scaled EM-ICP [10] share some properties with AICK. It does not assume one data association but rather consider a weighted combination of matches with the scale setting the weight.

Using key points (such as SIFT [11], SURF[12], BRIEF [13], BRISK[14] and FREAK [15]) typically extracted from the RGB information, reduces the need to treat all pixels and using feature descriptors allows for reliable associations. An example of using key points and ICP to register RGB images is given by [16]. In this work we use SURF and ORB [17] which extends BRIEF with invariance to rotation. Key points are often detected by FAST [18] or Harris corners [19].

The Kinect Fusion algorithm [20] uses a dense, non-parametric, representation for the reference frame from which an artificial point cloud is sampled and registered against.

A common data association problem is that of looking for a match between one frame and all frames previously seen. Finding these, so called, loop closures are key to a successful implementation of SLAM. Here the question is first if the two frames match at all and if so what the transformation is. Matching feature by feature in each frame is prohibitively slow. A common approach taken is to make use of visual vocabularies [21]. The basic idea is to form clusters in descriptor space and assign a label to each cluster or word. The discretisation of descriptors into words means that feature matching can be done by comparison two integer indices (the label of the word). This has laid the foundation for FAB-MAP [22] and its follow-ups.

A major part of registration is the problem of outlier rejection i.e. the fact that there may be regions with no overlap. Using a suitable model, RANSAC [5] can be used to separate inliers from outliers and calculate model parameters.

## III. THE AICK ALGORITHM

The AICK algorithm is an efficient and accurate way to register two frames of RGB-D data. It exploits keypoints that have both a 3D position in space as used by ICP and a descriptor which characterizes the surrounding context of the point. In contrast to ICP, it is able to find a good registration even when no initial guess is given. AICK is an iterative algorithm that adaptively changes from emphasizing the descriptor match to emphasizing the geometric fit between the points in the two frames. At the later stages it becomes essentially ICP but having avoided the local minima that result from incorrect initial matches. The results are thus as for ICP with less failures.

As said, in ICP one must start with an initial guess of the transformation between the two frames. One then finds all the matching pairs of points. The matching criteria is the smallest Euclidean distance, $d_e$ between the 3D points. After finding all matches where $d_e$ is below a threshold, the transformation is recomputed to minimize the sum of these distances.

The main strength of the ICP method is that it gives very accurate transformations when the matches are correct. It is most suitable for dense point clouds where sampling artifacts are not significant.

The main weakness of ICP is that if the initial guess leads to too many incorrect matches the solution can get 'stuck' trying to make those fit. It needs most of the initial matches to either be correct or at least on the correct smooth surfaces. The need to have a good guess to start with is rather problematic as it is just this transformation that we are after. It would be better if the method did not require any initial guess, especially when looking for loop closures. In AICK the initial match is independent of the transformation as it is based solely on the descriptor information.

AICK does not match dense point clouds but rather keypoints. Two similar features or the same feature seen from different angles will have descriptors that are close in this descriptor space. This way we reduce the number of points to consider for matching to only those points that have a key point associated with it. This then addresses the problem of which points to select as well.

AICK does the same two phases, match and optimize, as ICP but it uses a different matching metric which adapts over the course of the iterations. Instead of $d_e$ we use $d_i$,

$$d_i = (1 - \alpha^i)d_e + \alpha^i d_d, \tag{1}$$

where $i \in \{0, 1, 2, 3, \dots\}$ is the iteration number, $d_d$ is the distance, L2 norm, in descriptor space and the constant parameter $\alpha \in [0, 1]$ is the decay factor to move from pure descriptor distance, $(i = 0)$ to nearly only Euclidean distance, $(\alpha^i << 1)$

In addition to assessing what points are closest, the distance metric is also used to reject points that are too far away. The distance $d_e$ and $d_d$ have different units and finding a threshold for the combined distances requires some thought. We define this threshold according to

$$\lambda_i = (1 - \alpha^i)\lambda_e + \alpha^i \lambda_d, \tag{2}$$

where $\lambda_e$ is the outlier rejection corresponding to the euclidean distance and $\lambda_d$ corresponding to the feature distance.

### A. Non exhaustive search strategy

AICK reduces the computational requirement compared to standard ICP in several ways. Firstly, because it only uses points with an associated key point. Experiments also show that we do not have to perform an exhaustive search for the best matches. That is, even if we limit the search for the keypoints in one frame to only a small subset and miss some matches performance is maintained high given that we start with enough key points. This opens up ways to make the algorithm more efficient by trading off the expensive step of finding all the matches that fall below our threshold.

A common way to reduce the cost of matching, which we also make use of, is to use a so called 'vocabulary' of words do this we use the method of learning a 'vocabulary' of words

as in the bag of words method.[1] We learn the words using different data from what we test on. Learning corresponds to clustering the descriptors from all the training images into a predetermined number of clusters. The words are then the mean descriptors for each cluster.

With every keypoint, $p_k$, we associate a list of its closest words in that frame, which we denote as $\Psi(p_k)$. $\Psi(p_k)$ contains the words to which the descriptor distance of the keypoint is less than a threshold, $R_w$. This can be done swiftly if the vocabulary contains few clusters or if the words are arranged in a tree structure that speeds up this search. Note that this is only done once per frame, i.e., if we match the frames to many other frames we need not recompute $\Psi$. This is key for applications such as SLAM where detection of loop closures look at the same frame for matches several times. To look for matches for a key point in frame A to key points in frame B we start with the closest words in frame A and match the key point only to the key points associated with the same words in the other frame. This can be made very fast by creating an index per frame from words to keypoints. Instead of having to match all points to all points we only match each point to a (small) subset of the points in the other frame. This can speed up the expensive association step by an order of magnitude in most cases. We consider this a generalization of the original algorithm as using $R_w = \infty$ is equivalent to the original algorithm.

### B. Quality of registration

To assess the quality of the registration we subsample every RGB-D frame using a grid in the image plane. We store one validation point for each intersection point in this grid. When two frames are matched the validation points in the two frames are backprojected into the depth of the other frame.

These points are scored based on the difference $d_i$ between the backprojected depth and the measured depth. If the absolute value of $d_i$ is smaller than a threshold $\Gamma_{good}$ this point is considered valid. In order to make use of knowledge of open space between the sensor and the depth reading, validation points that end up much closer(quantified in the form of a threshold $\Gamma_{bad}$) to the sensor than the measured depth are penalized by assigning it a value $\delta < 0$. The definition of $score(d_i)$ is summarized in the following equation.

$$score(d_i) = \begin{cases} 1, & |d| < \Gamma_{good} \\ \delta < 0, & d > \Gamma_{bad} \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

The overall quality measure, $W$, is given

$$W = \frac{1}{M} \sum_{i=1}^{N} score(d_i) \quad (4)$$

where $N$ is the number of overlapping validation points and $M = max(M_{min}, N)$, with $M_{min}$ ensuring that $W$ becomes small when $N$ is small, i.e. when there is a small overlap.

[1]We do not use the 'bags' in this work only the words. The bags might be useful to chose which two frames to try to register to one another which is a question not addressed here.

## IV. EXPERIMENTAL SETUP

For evaluation we use [6] which is a publicly available dataset designed for the purpose of benchmarking RGB-D SLAM algorithms in realistic indoor environments. The dataset is complete with ground truth and contains sequences of RGB-D data captured using a Kinect. To be specific, we use the sequence *fr1/room* which at the time of writing this paper was the longest of all the sequences in the natural office environment subset. This data set is well suited to its designed purpose of testing state of the art registration algorithms in that the motion has all 6 degrees of freedom and the movement is both rapid and uneven.

### A. Performance Measure

We employed a performance measure provided together with the dataset [6]. The measure is based on the relative pose error, which is found by first transforming the origin pose using the estimated transformation and then transforming it back using the inverse of the ground truth transformation. In a perfect case without error, this results in a pose matching the origin pose.

$$E_i = G_i^{-1} Q_i - I, \quad (5)$$

where $G_i$ is the ground truth transformation for transformation $i$, $Q_i$ is the estimated transformation and $I$ is the identity matrix. We analyze the translation component of $E_i$ by measuring the relative distance between the pose obtained after the two transformations described above and the origin pose as suggested in [6]. This error will be given in meters, see (6) for mathematical formulation. As a means of summarizing the results for a set of translation errors we define *successratio* as the ratio of translation errors smaller than some threshold $\lambda_t$ in the set. That is the registration is considered a 'success' if it satisfies (6).

$$E_i^{Translation} = (\sum_{j=0}^{2} ||E_{i,j,3}||^2)^{1/2} < \lambda_t. \quad (6)$$

It is worth noting that when *successratio* $= 0.5$, $\lambda_t$ is the median error. Similarly to using the median error the *successratio* considers all outliers as equal, meaning that gross outliers does not bias the analysis. This formulation allows us to analyze the distribution of errors by varying the threshold $\lambda_t$.

### B. Algorithms tested

Three different registration algorithms in addition to AICK[2] were ran and compared on the test set. The parameters for the algorithms were optimized by hand by testing a large set of values to yield good performance within a maximum of roughly five minutes of execution time per pairwise registration.

*1) GICP:* We use the GICP implementation provided by the Point Cloud Library (PCL [23])[3].

[2]AICK using $\lambda_e = 0.01m$ and $\lambda_f = 0.2$.

[3]GICP was allowed to run for 25 iterations. Rejection threshold $= 0.004m$.

*2) 3D-NDT:* We use the 3D-NDT implementation provided by the Point Cloud Library (PCL [23])[4].

*3) 3-POINT RANSAC:* We used the RANSAC algorithm on this problem by first forming a list of potential matching keypoint pairs based on the similarity of the descriptors only. We then randomly select three of these pairs to define a transformation between the frames, which we will call the 'model'. We then count the number of 'inliers' according to the model. The model with the most inliers is chosen and updated by using all of the found inliers. In forming the list of potential matched pairs only associations between keypoints with descriptor distance $d_d \leq \lambda_f$ are used. Inliers are calculated by transforming the keypoints in one frame by the model and associating the transformed keypoints to the closest keypoint in the other frame. If the euclidean distance $d_e \leq \lambda_e$ between these keypoints the association is counted as an inlier[5]. For the 3-point RANSAC algorithm we use SURF keypoints.

We will use two different types of keypoints, SURF [12] and ORB [17]. The Surf keypoints will be extracted using OpenSURF Library[24]. Using our test set we found an average of 906 surf keypoints with valid depthdata in an average of 0.12 seconds. To extract the ORB keypoints we use OpenCV [25]. Using our test set we found an average of 857 ORB keypoints with valid depthdata in an average of 0.011 seconds.

### C. Experimental Procedure

The registration experiments were performed by estimating transformations between consecutive frames of the data sequence. In order to test robustness to larger transformations, we performed the experiments for pairs of frames separated by different lengths of time. Performance is measured quantitatively using the measure described in (6). The point clouds were created with calibrated camera parameters. In section V-C we visualize the effects of accumulating a sequence of consecutive frame transformations and transforming the appropriate pointclouds into a common coordinate frame.

## V. EXPERIMENTAL RESULTS

We plot the *successratio* versus a varying $\lambda_t$ for (6) up to 0.05 meters using consecutive frames (around 30ms apart) for the different algorithms in fig. (1)[6] . This allows us to see both the size and variation of the translation error of the different methods when the transformation between frames is relatively small. A steep curve can be interpreted as good performance as that would mean that the method often yields a transformation with a small translation error. One sees that, for consecutive frames, all of the methods reach nearly 100% *successratio* at a relatively small $\lambda_t$. The conclusion is

[4]To keep the runtime reasonably low the pointclouds were subsampled through the use of a voxelgrid with a voxel size of $0.02m$. 3D-NDT was allowed to run for 25 iterations, with $resolution = 0.1$ and $stepsize = 0.09$.

[5]We iterated the RANSAC over 400 random models in searching for the best model using $\lambda_e = 0.02m$ and $\lambda_f = 0.2$.

[6] AICK was run for 25 iterations with $\alpha = 0.8$ and $R_w = \infty$

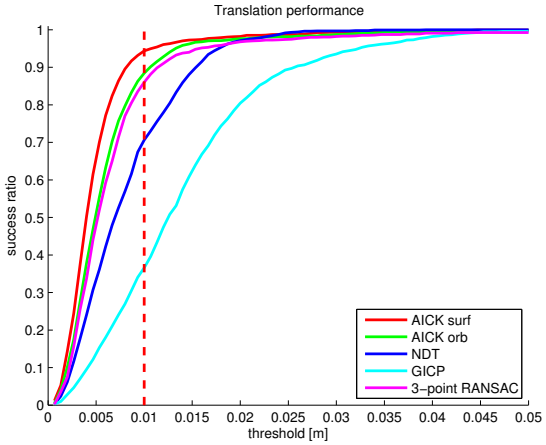


Fig. 1. The *successratio* as a function of the threshold on the translation error in m. Here we use all the found keypoints. The red dashed line shows the threshold used in fig. (2). Meaning that the intersections with the red dashed line are equivalent to values for the *successratio* in fig. (2) when the time difference between frames equals 30 ms.

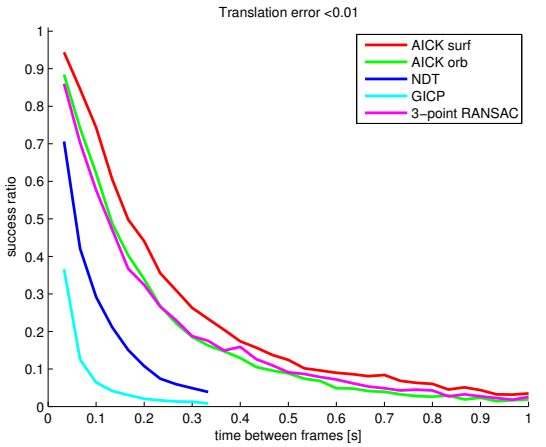


Fig. 2. *successratio* as a function of the time difference between frames with a fixed threshold on the translation error of 0.01 m. Here we use all the found keypoints.

that while AICK using surf keypoints outperforms the other methods in this test all of the methods are fairly accurate given small displacements of the camera. It is also interesting to note that the difference between the use of surf and orb keypoints is relatively small for AICK.

It is also informative to see the result on the *successratio* by using a fixed threshold and varying the time difference between the frames being matched. This is shown in fig. (2)[6] for a threshold of 0.01 meters. It is clear that the AICK and 3-point RANSAC degrades much slower than GICP and NDT when the camera displacement increases.

### A. Quality of registration

By rejecting bad transformations we can ensure a higher performance for the non-rejected transformations. Fig. (3)[7] shows the effects on the *successratio* for two different thresholds on $W$ (see eq. (4)). Notice the large difference

[7] Using a 10-by-10 subsampling grid with a minimum overlap of 500 samples, $\Gamma_{good} = 0.01m$, $\Gamma_{bad} = 0.075m$ and $\delta = -2$.

| Algorithm | | | Iterations | Avg runtime [s] | *successratio* for threshold $\lambda_t$ | | |
|---|---|---|---|---|---|---|---|
| | **Keypoints** | $\mathbf{R}_w$ | | | $\lambda_t = 0.0033$ | $\lambda_t = 0.01$ | $\lambda_t = 0.05$ |
| AICK | on avarage 906 surf keypoints | $\infty$ | 25 | 0.180 | 0.374 | 0.944 | 0.993 |
| AICK | on avarage 857 orb keypoints | $\infty$ | 25 | 0.135 | 0.276 | 0.885 | 0.999 |
| AICK | max 200 surf keypoints | $\infty$ | 5 | 0.00385 | 0.281 | 0.902 | 0.993 |
| AICK | max 350 orb keypoints | $\infty$ | 10 | 0.0113 | 0.209 | 0.833 | 0.998 |
| AICK | max 200 surf keypoints | 0.26 | 5 | 0.000445 | 0.258 | 0.888 | 0.992 |
| AICK | max 200 surf keypoints $W > 0.7$ | 0.26 | 5 | 0.000480 | 0.313 | 0.953 | 0.999 |
| AICK | max 200 surf keypoints $W \leq 0.7$ | 0.26 | 5 | 0.000480 | 0.156 | 0.740 | 0.977 |
| AICK | max 200 surf keypoints $W > 0.5$ | 0.26 | 5 | 0.000480 | 0.285 | 0.931 | 0.999 |
| AICK | max 200 surf keypoints $W \leq 0.5$ | 0.26 | 5 | 0.000480 | 0.075 | 0.500 | 0.932 |
| AICK | max 350 orb keypoints | 0.165 | 10 | 0.000717 | 0.209 | 0.828 | 0.995 |
| GICP | | | 25 | 224 | 0.070 | 0.366 | 0.996 |
| NDT | | | 25 | 237 | 0.177 | 0.706 | 1 |
| 3-point ransac | | | 400 | 4.09 | 0.255 | 0.860 | 0.993 |

TABLE I

RUNTIME COSTS AND PERFORMANCES FOR THE TESTED ALGORITHMS. $R_w$ IS THE RADIUS AROUND THE KEYPOINT TO FIND MATCHING WORDS.

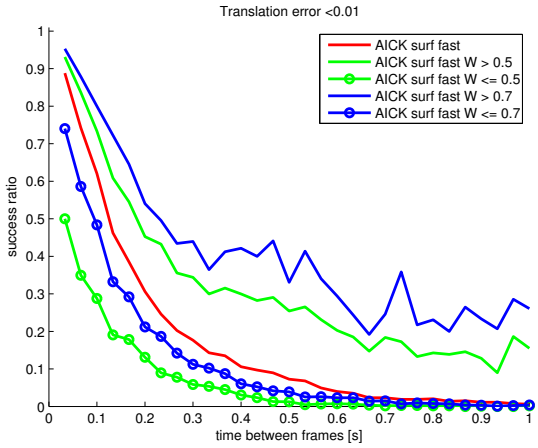


Fig. 3. *successratio* as a function of the time difference between frames with a fixed threshold on the translation error of 0.01 m.

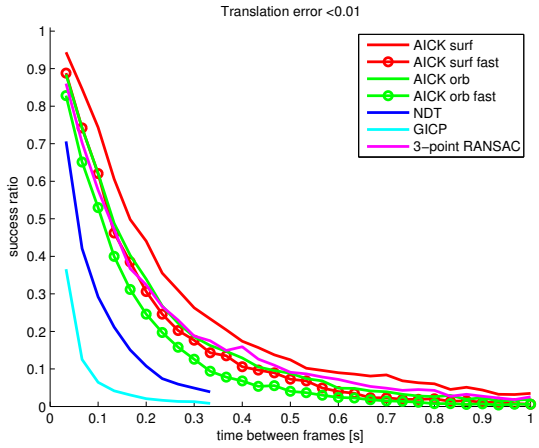


Fig. 4. *successratio* as a function of the time difference between frames with a fixed threshold on the translation error of 0.01 m.

between the cases when the $W$ is over a specified threshold as compared to being under this threshold.

### B. Runtime

We can control the runtime to performance trade-off of the algorithm using three main parameters: the number of keypoints used, the number of iterations the algorithm is allowed to run and the threhold $R_w$. The effects on the *successratio* from limiting these parameters can be seen in Table I for registration of two consecutive views. The cost for extracting keypoints used by AICK or 3-point RANSAC is not included in the table. The reason being that in many applications keypoint extratction is only done once per frame whereas frame to frame registration may be run multiple times per frame. For the frames in the test set we found an average of 906 surf keypoints with valid depthdata in an average of 0.12 seconds and an average of 857 ORB keypoints with valid depthdata in an average of 0.011 seconds. It can be seen that the keypoint based methods are much faster than the non-keypoint based methods. Obviously runtime is dependent on implementation but since the keypoint methods deal with a lot less data there are less calculations to be done. By controlling the parameters for the AICK algorithm results similar to that of the 3-point RANSAC can be achieved in a fraction of the time. It can also be seen in table (I) that extracting an estimate of the quality of the registration can be done at a small computational load.

The performance of AICK using different parameter settings is shown in table (I) and in fig. (4)[8,9]. $R_w = \infty$ indicates not using words at all. Table (I) shows that tuning the parameters of the algorithm can greatly speed up the registration while fig. (4)[8,9] shows that the drop in performance was relatively small.

### C. Visual inspection

The AICK algorithm clearly outperforms the other methods in both robustness and precision as the above results show. In fig. (5) we visualize the results of accumulating transformations estimated by AICK over a sequence of 1000 frames. This is a common and effective way to allow for a qualitative evaluation by visual inspection. Because

[8]AICK orb fast was run 10 iterations with $\alpha = 0.6$, a maximum of 350 orb keypoints and $R_w = 0.165$.

[9]AICK surf fast was run 5 iterations with $\alpha = 0.3$, a maximum of 200 surf keypoints and $R_w = 0.26$.

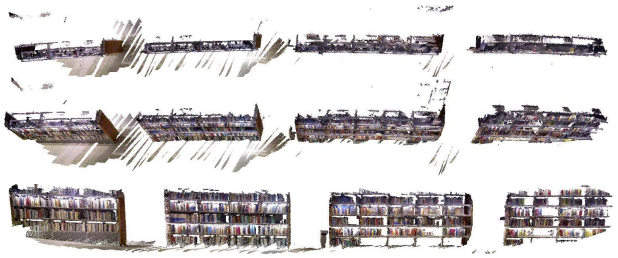

Fig. 5. Rendering of the the points given by frame-to-frame transformation estimates when walking past a series of bookshelfs in the KTH library. The data is displayed from three different view points. The bookshelves are lined up in the library and the upper part of the image shows that our method produces results very close to this even using pure dead-reckoning.

transformations are added frame by frame, i.e. pure dead-reckoning, errors, especially in orientation, will result in clearly visible distortions. To remove the background and avoid displaying noisy data, only data captured close to the sensor is displayed. The absence of distortions lends credibility to the practical use of the AICK method on real world systems.

## VI. SUMMARY AND CONCLUSIONS

In this paper we have clearly shown that AICK is natural choice for small robots, mobile devices and other embedded systems with limited resources but where high performance is needed. This is made possible by transitioning between coarse, appearance-based registration such that no initial estimate is needed and fine registration using position-based ICP on distinctive keypoints. In order to verify the performance of our method, we employed a standard benchmark consisting of a dataset and performance measure [6]. We compared the method to three different high performance registration techniques. In the experiments our method showed a significant improvement of both robustness to larger transformations and precision of the final result which can be attributed to the adaptive distance metric. Furthermore, sub-sampling of the point cloud into a selection of keypoints resulted in an algorithm orders of magnitudes faster than algorithms used for comparison.

## ACKNOWLEDGMENT

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement No 600623 and by SSF through its Centre for Autonomous Systems.

## REFERENCES

[1] J. Ekekrantz, A. Pronobis, J. Folkesson, and P. Jensfelt, "Adaptive iterative closest keypoint," in *Proceedings of the 6th European Conference on Mobile Robots (ECMR '13)*, 2013.

[2] P. Besl and N. McKay, "A method for registration of 3-d shapes," *IEEE Trans. on Pattern Analysis and Machine Intel.*, no. 2, pp. 239–256, 1992.

[3] A. Segal, D. Haehnel, and S. Thrun, "Generalized-icp," in *Proceedings of Robotics: Science and Systems*, (Seattle, USA), June 2009.

[4] M. Magnusson, A. Lilienthal, and T. Duckett, "Scan registration for autonomous mining vehicles using 3d-ndt," *Journal of Field Robotics*, vol. 24, pp. 803–827, 2007.

[5] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[6] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[7] Y. Chen. and G. Medioni, "Object modeling by registration of multiple range images," in *Proc. of the 1992 IEEE Intl. Conf. on Robotics and Automation*, pp. 2724–2729, 1992.

[8] T. Stoyanov, M. Magnusson, and A. J. Lilienthal, "Point Set Registration through Minimization of the L2 Distance between 3D-NDT Models," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 14–19 2012.

[9] L. Montesano, J. Minguez, and L. Montano, "Probabilistic scan matching for motion estimation in unstructured environments," in *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, 2005.

[10] S. Granger and X. Pennec, "Multi-scale em-icp: A fast and robust approach for surface registration," in *European Conference on Computer Vision, 2002*, pp. 418–432, 2002.

[11] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[12] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Computer Vision–ECCV 2006*, pp. 404–417, Springer, 2006.

[13] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: binary robust independent elementary features," in *Computer Vision–ECCV 2010*, pp. 778–792, Springer, 2010.

[14] S. Leutenegger, M. Chli, and R. Y. Siegwart, "Brisk: Binary robust invariant scalable keypoints," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 2548–2555, IEEE, 2011.

[15] A. Alahi, R. Ortiz, and P. Vandergheynst, "Freak: Fast retina keypoint," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 510–517, IEEE, 2012.

[16] G. Yang, C. V. Stewart, M. Sofka, and C.-L. Tsai, "Registration of challenging image pairs: Initialization, estimation, and decision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, pp. 1973–1989, 2007.

[17] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: an efficient alternative to sift or surf," in *Computer Vision (ICCV), 2011 IEEE International Conference on*, pp. 2564–2571, IEEE, 2011.

[18] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Computer Vision–ECCV 2006*, pp. 430–443, Springer, 2006.

[19] C. Harris and M. Stephens, "A combined corner and edge detector," in *In Proc. of Fourth Alvey Vision Conference*, pp. 147–151, 1988.

[20] R. Newcombe, A. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *Mixed and Augmented Reality (ISMAR), 2011 10th IEEE International Symposium on*, pp. 127–136, IEEE, 2011.

[21] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pp. 1470–1477, IEEE, 2003.

[22] M. Cummins and P. Newman, "FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance," *The International Journal of Robotics Research*, vol. 27, no. 6, pp. 647–665, 2008.

[23] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *International Conference on Robotics and Automation*, (Shanghai, China), 2011 2011.

[24] C. Evans, "Notes on the opensurf library," Tech. Rep. CSTR-09-001, University of Bristol, January 2009.

[25] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.