Abstract—We propose to overcome a significant limitation of the ICP algorithm used by KinectFusion, namely, its sole reliance upon geometric information. Our approach uses both geometric and color information in a direct manner that uses all the data in order to accurately estimate camera pose. Data association is performed by Lucas-Kanade to compute an affine warp between the color images associated with two RGBD point clouds. A subsequent step then estimates the Euclidean transformation between the point clouds using either a point-to-point or point-to-plane error metric, with a novel method based on a normal covariance test for automatically selecting between them. Together, Lucas-Kanade data association (LKDA) with covariance testing enables robust camera tracking through areas of low geometric features, without sacrificing accuracy in environments in which the existing ICP technique succeeds. Experimental results on several publicly available datasets demonstrate the improved performance both qualitatively and quantitatively.

Index Terms—Kinect, KinectFusion, ICP, Lucas-Kanade, Camera Tracking, Mapping

I. INTRODUCTION

THREE dimensional (3D) reconstruction of an environment is an important problem in robotics that has received much attention in recent years. The advent of inexpensive RGBD sensing (e.g., the Kinect sensor) has greatly improved the fidelity and speed with which such reconstructions can be made. These capabilities have far-reaching implications for the exploration and navigation of unknown environments, as well as for the manipulation of objects in those environments.

Of the many 3D modeling techniques that have been developed, the landmark KinectFusion method [19], [13] has established itself as perhaps the most accurate, real-time, dense modeling system using an inexpensive sensor. Recent work by others [29] has extended this algorithm to operate over large-scale environments. However, both the standard and the extended version use geometric information alone to align the camera with the model, thus requiring the environment to contain sufficient geometric features in order for the iterative closest point (ICP) algorithm to accurately estimate the camera pose. This deficiency was noted in [29], where preliminary experiments were conducted to explore the advantage of replacing the ICP algorithm with the output of a feature-based visual odometry system called FOVIS [11] to provide a more stable estimate of camera pose. The frame-to-frame approach of FOVIS, however, loses one of the key advantages of KinectFusion, namely the reduced drift that results from matching the current image to the model rather than to the previous image.

In this paper we present an approach to RGBD camera tracking in environments with low geometric information. The proposed approach uses all the data (both color and depth), preserves the advantages of KinectFusion’s image-to-model matching, and obviates the need for the feature point extraction and feature correspondence steps inherent in feature-based visual odometry systems. The key to the approach is to replace the projective data association (PDA) point matching algorithm [3] used by KinectFusion with a data association technique driven by Lucas-Kanade [18], [26], [2]. We introduced this matching algorithm, which we call Lucas-Kanade data association (LKDA), in [21]. In this paper we extend that work with an automatic method for selecting the appropriate error metric based on the geometry of the scene, a GPU implementation of the matching, and more thorough experimental results to demonstrate the improvement that results from LKDA as well as to compare it with alternative techniques. These results on standard datasets demonstrate that LKDA combined with the automatic error metric enable camera tracking to succeed in areas of low geometry, without sacrificing either computational efficiency or accurate camera tracking in highly geometric environments.\(^1\)

II. PREVIOUS WORK

Before the advent of recent inexpensive RGBD sensors, several researchers addressed the problem of improving ICP by incorporating color information. Both geometric and color information are incorporated into the distance metric used by ICP in a number of systems [5], [12], [14]. Similarly, Druon et al. [6] segment colored point clouds based on hue, then perform ICP while requiring matching points to belong to the same color class.

More recently, a popular way to use visual information to improve alignment is to match feature points, then apply RANSAC to the features with depth values. This is the approach employed by Henry et al. [10], in which the visual feature associations from RANSAC are then combined with dense point associations in an ICP framework. This same procedure of feature matching followed by RANSAC is adopted in both the FOVIS visual odometry system [11] and the algorithm of

\(^1\)Although our LKDA approach will fail when there is no visual texture (e.g., a white wall), such scenes have low geometric variation as well, since geometry usually generates visual texture via shading and shadows.
Engelhard et al. [8], the latter of which also employs ICP and is evaluated in [7]. As expected, the primary drawback of these techniques is the failure of feature detection and matching when the scene contains little photometric texture.

An approach that is closely related to our own is to register RGBD images via direct image warping. Tykkälä et al. [27] formulate the 3D registration of colored point clouds as a direct minimization of both depth and color information. To reduce the computational load, in followup work the same researchers [1] minimize only the image intensities, without using depth. In related work, Steinbrücker et al. [24] align two colored point clouds by employing Lie algebra to solve for the twist coordinates representing the 3D transformation that maximizes the photocoherence of the projections. This work has been extended by Kerl et al. [16] to achieve real-time performance (30 Hz) on a single CPU, as well as to combine both photocoherence and depth consistency in the minimization process [15]. Another direct approach is DTAM [20], which uses dense methods for whole image alignment without a depth sensor but stores the model as a set of keyframes rather than as a dense volume.

Other researchers have focused on extending KinectFusion. Henry et al. [9] propose a system consisting of local patch volumes, which are similar to KinectFusion volumes, connected together in a graph optimization framework to ensure global consistency. In another approach that generates large maps, Whelan et al. [28] combine KinectFusion’s ICP algorithm with a GPU implementation of the RGBD alignment approach of [24], and they provide a switching mechanism between the resulting estimate and that of the feature-based FOVIS. As expected, they show that both FOVIS and RGBD alignment fail when the scene contains little photometric texture, KinectFusion’s ICP fails when there is little geometric texture, and a combination of these methods yields improved tracking robustness.

Like the work of Whelan et al. [28], we focus upon improving the camera alignment step of KinectFusion by considering both visual and depth information. However, instead of combining off-the-shelf algorithms with a switching mechanism between them, we derive a novel algorithm that first aligns the intensity images then uses the resulting correspondences to align the depth images with an automatic selection of either point-to-point or point-to-plane error metrics based on the scene geometry. In Section V we show that our relatively straightforward, direct approach produces more consistent results than those obtained by more complicated methods, without sacrificing computational speed.

III. LUCAS-KANADE DATA ASSOCIATION

Let \( P = \{p_i\}_{i=1}^N \) and \( Q = \{q_j\}_{j=1}^N \), where \( p_i, q_j \in \mathbb{R}^3 \), be two point clouds in 3D Euclidean space. The goal of data association is to compute a function \( f \) that maps indices in one point cloud to those of the other, so that \( f(i) = j \) indicates a corresponding pair of points \( p_i \leftrightarrow q_j \). Let \( I_P \) and \( I_Q \) be the current and reference RGB images, respectively, associated with the point clouds, and \( d_P \) and \( d_Q \) the current and reference depth maps. In our implementation \( I_Q \) is the previous image, while \( d_Q \) arises from projecting the truncated signed distance function (TSDF) according to the previous camera pose.

![Fig. 1. LEFT: PDA establishes correspondence between two point clouds, \( P \) and \( Q \), by projecting each point \( p_i \in P \) of one cloud onto the other’s depth camera, \( D_Q \), to find the nearest projection of \( q_j \in Q \). RIGHT: LKDA establishes correspondence by projecting each point \( p_i \) onto the RGB camera \( C_P \), then warping the projected point onto the other RGB camera \( C_Q \) using the estimated warp from Lucas-Kanade to find the nearest projection of \( q_j \). (The normals \( n_j \) are used in the point-to-plane error metric.)](image)

Projected data association (PDA) [17] establishes correspondence between \( P \) and \( Q \) by projecting the points from both point clouds onto the same image plane, then considering points that project onto the same pixel to be corresponding points. See Figure 1. In a similar manner, Lucas-Kanade data association (LKDA) also finds the closest point on the image plane, but only after first transforming the projected coordinates according to the warp function found by Lucas-Kanade:

\[
 f(i) = \arg\min_j \| \varphi(C_Q q_j) - \varphi(W(C_P p_i; \zeta)) \|,
\]

where \( \zeta \) contains the warp parameters, \( C_P \) and \( C_Q \) are \( 4 \times 4 \) homogeneous matrices describing the 3D position and orientation of the RGB cameras, the tilde indicates homogeneous coordinates, and \( \varphi \) dehomogenizes the coordinates.

The PDA approach relies solely upon geometric information, that is, the spatial coordinates of the points in the two clouds. As a result, when the environment does not provide sufficient geometric texture (e.g., the camera is looking at a planar wall), or when most of the depth readings are invalid due to the limited range of the sensor or occur outside the TSDF maintained by KinectFusion (e.g., the camera looks down a hallway), PDA will fail to provide enough correspondences to track the camera. The LKDA approach improves the alignment of point clouds in such environments, since the photometric information is not affected by the scene geometry, the range of the sensor, or the size of the TSDF.

The Lucas-Kanade algorithm is a differential method for computing the optical flow of an image by finding the parameters \( \zeta \) that minimize the sum of squared distance error

\[
\epsilon_LK = \sum_x \left( I_Q(W^{-1}(x; \zeta)) - I_P(x) \right)^2,
\]

where the summation is over all the pixels \( x = [x \ y]^T \) in the image, and \( W(x; \zeta) \) is a parametric warp function that brings the two images into alignment. This error is minimized by linearizing about the current estimate and repeatedly solving
the following linear system
\[
\left( \sum_x g g^T \right) \Delta \zeta = \sum_x g \left( I_Q(W^{-1}(x; \zeta)) - I_P(x) \right)
\]  
for \( \Delta \zeta \), where \( g = \left( \frac{\partial W}{\partial \zeta} \right)^T \nabla I_P \), and \( \nabla I_Q \) contains the gradient of \( I_P \). For efficiency, we use the inverse compositional algorithm [2].

Common choices for the warp function are translation, affine, or projective. Since we are interested in a global, featureless mapping technique that warps an entire image into another, the translation model is not expressive enough; either an affine or projective warp is thus required. We have found little difference in accuracy between these two alternatives (see the experimental results), but the projective warp incurs a substantial increase in computation due to the fact that the Hessian is dependent upon the current state of the system and therefore must be recomputed each iteration. As a result we use an affine warp, which is given by
\[
W(x, y; \zeta) = \begin{bmatrix} x(r_{xx} + 1) + y(r_{xy} + a_x) & x(r_{xy} + 1) + y(r_{yy} + a_y) \end{bmatrix},
\]
where \( \zeta = [r_{xx} \quad r_{xy} \quad r_{yx} \quad r_{yy} \quad a_x \quad a_y]^T \).

IV. ESTIMATING THE CAMERA POSE

Each iteration of ICP can be broken into six steps, and different choices within these steps lead to different variations of the algorithm [22]. Figure 2 illustrates these steps within the traditional ICP approach used by KinectFusion, compared with our LKDA approach. Instead of recomputing data association after each alignment, we first perform data association, then perform alignment. This decoupling, which is enabled by our separate use of RGB and depth images, leads to significant speedup without sacrificing robustness. At the same time, the correspondences used in each iteration of the algorithm are not necessarily the same, because the rejection step retains only a subset of them. In this section we detail the steps that are different between our approach and KinectFusion.

A. Point Matching

As already mentioned, projective data association (PDA) establishes correspondence by finding the closest point from the other cloud as projected onto the image plane. In a similar manner, Lucas-Kanade data association (LKDA) also finds the closest point on the image plane, but only after first transforming the projected coordinates according to the warp function found by Lucas-Kanade. Unlike traditional ICP, however, the LKDA point matching step is performed only once per pair of image frames, and subsets of the correspondences found in this step are then used in the iterations of the alignment process.

B. Outlier rejection

As in [19], our approach rejects point correspondences that exceed a distance or angular threshold. In addition, corresponding points whose color difference exceeds a predefined color threshold \( \rho \) are rejected:
\[
||I_Q(x, y) - I_P(W(x, y; \zeta))|| > \rho.
\]
Note that the rejected points in each iteration of the alignment process are not necessarily the same, so that a different subset of points contribute to the error metric in each iteration. In our implementation the distance, angular, and color thresholds are 0.1 m, 20°, and \( \rho = 25 \), respectively.

C. Automatic Selection of the Error Metric

Two common choices of the metric used for alignment are point-to-point and point-to-plane [4]. If there is sufficient geometric texture in the environment, then the latter converges faster than the former [22], which explains its adoption by KinectFusion. On the other hand, if all point correspondences are on a flat wall then the point-to-plane error metric will not result in correct alignment, because there will be no mechanism to induce a lateral shift between the point clouds in the direction perpendicular to the normals. See Figure 3.

To overcome this limitation we use a linear combination of the 3D Euclidean transformations computed using the point-to-point \( T_{pp} \) and the point-to-plane \( T_{pp} \) error metric:
\[
T = (1 - \lambda)T_{pp} + \lambda T_{pp},
\]
where \( 0 \leq \lambda \leq 1 \) is a weighting factor. The value of \( \lambda \) is determined automatically at run time by solving for the condition number \( \kappa \) of the covariance matrix of the geometric normals seen by the current frame. In areas of low geometry the value of \( \gamma = \kappa^{-1} \) is low, while in areas of high geometry \( \gamma \) is high. We model \( \lambda \) as a sigmoid
\[
\lambda = \left(1 + e^{-\alpha(\gamma - \overline{\gamma})}\right)^{-1},
\]
where $\alpha$ is a large constant (determined experimentally to be 300), and $\hat{\gamma}$ indicates the value of $\gamma$ for which the two metrics are weighted equally, i.e., $\lambda = 0.5$. To determine $\hat{\gamma}$, for each frame over several sequences we computed $T_{pp}$, $T_{pp}$, and $\kappa^{-1}$, comparing the former two with the ground truth. The resulting scatter plot can be seen in Figure 4, along with the intersection of robust exponential curves fit to both data sets, yielding $\hat{\gamma} = 0.01$. Figure 5 shows the results of both metrics on two scenes, illustrating that $T_{pp}$ yields less error on scenes with low geometry, while $T_{pp}$ tends to perform slightly better on scenes with high geometry.

**V. EXPERIMENTAL RESULTS**

To implement the proposed approach, we modified the open-source version of KinectFusion known as KinFu in the PCL library [23] with our own C++ and GPGPU code. In our first set of experiments, we compared the unmodified PDA version with six different versions of LKDA using either grayscale or color images, and either a translation, affine, or projective warp. To facilitate quantitative comparison, we used the data from the RGB-D SLAM Dataset and Benchmark [25], which provides synchronized depth and color data from a Kinect or Xtion sensor as well as the ground truth sensor trajectory from a high-accuracy motion capture system. For these experiments, we used 5 different datasets exhibiting a variety of small environments, which can roughly be categorized as follows:

- **high-geometry**: The environment contains a large amount of geometric variation throughout, enabling geometry-based sensor tracking. (fr1/xyz, fr1/door, fr1/desk2)
- **planar**: The environment is largely planar (that is, without much geometric variation), making geometry-based sensor tracking difficult. (fr3/ntnw)
- **out-of-bounds**: For at least some frames of the sequence, the sensor is located so that a majority of the depth values are outside the TSDF volume. (fr1/rpy)

We used a $3 \times 3 \times 3$ m TSDF volume divided into 512$^3$ voxels (each voxel is 5.86 mm per side).

The results on these 5 datasets are shown in Table I, where the RMSE translational drift (in meters per second) and RMSE rotational drift (in radians per second) are displayed. The drift is also known as the Relative Pose Error (RPE) [25]. For the high-geometry sequences, all the methods succeed, and there is little difference in accuracy in either position or orientation. For the planar (fr3/ntnw) or out-of-bounds (fr1/rpy) sequences, however, the standard KinectFusion algorithm fails to maintain sensor tracking. There is a noticeable improvement using affine over translation, but almost none by using projective over affine. To save space, we show only the grayscale results, since the color results are nearly identical.

We directly compared our method with the approach of Whelan et al. [28] and the Dense Visual Odometry (DVO) algorithm of Kerl et al. [15], [16]. Results of these comparisons are displayed in Tables II and III, respectively. While the algorithm of [28] performs better on a frame-to-frame basis, our approach yields a better overall estimated trajectory, leading

![Fig. 3. When the scene is planar, the point-to-plane error metric fails to correctly align point clouds, even with perfect correspondence.](image)

![Fig. 4. Scatter plot of $\gamma$, the inverse of the condition number of the covariance matrix of the normals, versus the error of the estimated translation (averaged over all frames of several sequences). The red dots indicate the results using the point-to-plane error metric ($T_{pp}$) to estimate odometry, while the blue dots show the results from the point-to-point error metric ($T_{pp}$). Robust exponential curves were fit to both data sets to determine their intersection.](image)

![Fig. 5. Cumulative camera translation error for both metrics when reconstructing a relatively flat environment (left) versus a geometrically rich environment (right).](image)

**D. Error minimization**

Given the correspondences between the two image frames that survived the rejection step, along with the value $\lambda$, the two point clouds can be aligned. The transformation $T_{pp}$ is computed in closed-form using orthogonal Procrustes analysis, while $T_{pp}$ is computed iteratively as described in [17]. The camera pose is then given by (6).

\[
\begin{align*}
\mathbf{T}_{pp} &= \mathbf{R}_{pp} \mathbf{t}_{pp} \\
\mathbf{R}_{pp} &= \mathbf{K} \mathbf{M} \mathbf{K}^{-1}, \\
\mathbf{M} &= \frac{1}{\lambda} \left( \mathbf{I} + \frac{1}{\kappa} \mathbf{Q} \right), \\
\mathbf{t}_{pp} &= \mathbf{K}^{-1} \mathbf{Q} \mathbf{t}, \\
\mathbf{Q} &= \left( \mathbf{I} - \frac{1}{\kappa} \mathbf{M} \right) \mathbf{M}^{-1}.
\end{align*}
\]
to lower absolute trajectory errors. Compared with [15], our approach is competitive, yielding the lowest error on several sequences.

Plots of the estimated camera position versus frame number for the planar (fr3/ntnw) and out-of-bounds (fr1/rpy) environments for the different algorithms are shown in Figure 6. Notice that KinectFusion with PDA is unable to maintain tracking in such environments, while our LKDA algorithm (grayscale affine) and the approaches of [28] and [15] are able to maintain camera tracking. Other variations of LKDA also perform well, except that the translation motion model is insufficient for the planar scene. Reconstructions of the out-of-bounds environment by rendering the point cloud from two approaches are shown in Figure 7 for comparison. Notice that LKDA achieves an accurate reconstruction of the scene, while the reconstruction from PDA is difficult to interpret.

The importance of the automatic error selection is shown in Table IV. Seven sequences were run using either a point-to-point or point-to-plane error metric, or with automatic selection between the two. As expected, the point-to-point error metric performed better in environments with little geometrical structure, while the point-to-plane error metric always yielded less error on sequences with high geometrical structure. The automatic selection mechanism enabled our approach to yield the minimum error on all sequences.

Timing comparisons of the different variations of our algorithm were conducted on several machines with varying computational power and GPUs. For standard KinectFusion
with PDA, nearly all the computation (approximately 85\%) is devoted to the alignment, with the remaining computation required for PDA. Our approach retains the alignment computation but replaces PDA with a more expensive LKDA step. On average, the ratio of the time taken by LKDA to PDA is 3:1 (grayscale translation), 6:1 (grayscale affine), and 45:1 (grayscale projective). (Color versions require nearly 3 times as much computation.) Therefore, our grayscale affine approach (the recommended version) is about 1.8 times as expensive as KinectFusion (since \(0.85 + 0.15(6) = 1.8\)). On a machine capable of running standard KinectFusion at 30 Hz, our algorithm runs at approximately 17 Hz; on a machine that runs KinectFusion at 100 Hz, our algorithm runs at nearly 60 Hz.

VI. CONCLUSION

In this paper we have presented a direct technique to robustly estimate camera motion using an RGBD sensor, even when there is little geometric information in the scene. The approach retains the drift-free advantages of the match-to-model approach of KinectFusion, without requiring the estimation of visual features. The approach operates by incrementally computing the affine warp between successive images using Lucas-Kanade, yielding a correspondence between point clouds. This approach to correspondence, which we call LKDA, replaces the PDA technique implemented in the variant of ICP found in KinectFusion. In contrast with PDA, LKDA was shown on a number of publicly available sequences that it is able to maintain camera tracking in environments with limited geometric structure, as well as in environments that extend beyond the TSDF volume or contain depths outside the range of the sensor. Moreover, we have proposed an automatic mechanism for selecting between point-to-point or point-to-plane error metrics that properly handles a variety of environments. Future work will be aimed at extending this approach to large-scale environments.

VII. ACKNOWLEDGMENTS

The authors thank T. Whelan and C. Kerl for providing their data for comparison, and the anonymous reviewers for their helpful comments. This work was partially supported by NSF grant IIS-1017007.

REFERENCES


