Long Range RGBD and IMMU Odometry

Alexander Brett Bunting[1] and Ali Haydar Göktoğan[2]

***Abstract*— Localization in the absence of fixed external references is an ongoing challenge in the field of mobile robotics. Accurate positioning is essential for autonomous platforms to carry out their functions, however in GPS-denied environments or unknown unstructured terrain pose estimates from wheel encoders or inertial sensors can quickly accumulate significant errors. This paper presents a system in which color and depth (RGBD) odometry is fused with inertial and magnetic (IMMU) orientation data for long range, outdoor motion estimation. Experimental results are presented which demonstrate the effectiveness of the system over a variety of outdoor terrain environments over distances of hundreds of meters with drifts of between 8-25mm/sec. The data is captured using a low-cost IMMU, and a Kinect for Windows v2 RGBD camera physically shielded against interference from sunlight.**

I. INTRODUCTION

Localization is an important part of the guidance, navigation and control of autonomous mobile robotics. A common technique is to use external infrastructure as a reference, such as GPS, laser reflectors or sonar beacons [1]. However these are not always available, such as in unknown or GPS-denied environments, and localization must be performed using only data which can be captured with onboard sensors. Visual odometry estimates incremental pose changes by aligning image data from visual sensors [2], while inertial measurement units can estimate motion by integrating acceleration and gyroscopic data [3]. However these methods suffer from pose estimation errors which grow with time. The challenge is to maintain a high degree of accuracy so the accumulated error does not become intractably large.

This paper proposes an odometry system which fuses combined Red, Green, Blue and Depth (RGBD) visual odometry with orientation data from an Attitude Heading Reference System (AHRS) computed using a low-cost Inertial and Magnetic Measurement Unit (IMMU). Our odometry system combines 3D keypoint tracking, illustrated in Fig. 1, with iterative closest point (ICP) dense point cloud alignment, taking inspiration from the RGBD-ICP algorithm of Henry *et al.* [4]. The system is demonstrated over distances of hundreds of meters in a variety of unstructured outdoor environments. RGBD data is captured using the Kinect for Windows v2 sensor. We also present GPU accelerated implementations of several key odometry processing steps and show improvements in speed by a factor of 2-60.

[1]Alexander Brett Bunting is with The School of Aerospace, Mechanical and Mechatronic Engineering, The University of Sydney, NSW-2006, Australia alex.bunting314@gmail.com

[2]Ali Haydar Göktoğan is with the Australian Centre for Field Robotics (ACFR), The School of Aerospace, Mechanical and Mechatronic Engineering, The University of Sydney, NSW-2006, Australia a.goktogan@acfr.usyd.edu.au

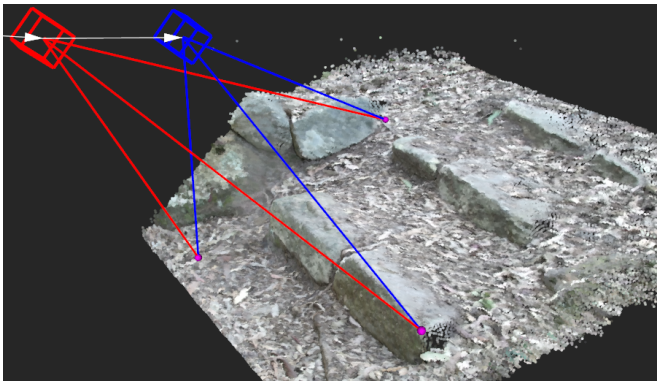

Fig. 1. Camera odometry estimation by tracking 3D keypoints

The primary contributions of this work are the implementation of a robust outdoor RGBD odometry system, an Extended Kalman Filter (EKF) for fusing AHRS orientation and RGBD odometry, a hardware setup that allows the Kinect to be used in direct sunlight, and the presentation of experimental results in outdoor long-distance trials.

II. RELATED WORK

The availability of dense color and depth information has led to a number of techniques being proposed which use RGBD cameras for odometry. Huang *et al.* [5] estimate the pose change between images by aligning visual keypoints projected to 3D using the depth image. Inlier detection and bundle adjustment are used to increase robustness. The depth image alone is used in the KinectFusion 3D scanning package [6], where point clouds are projected from the depth image and aligned using a modified version of ICP [7]. Kerl *et al.* in their Dense Visual Odometry system [8] use the entire depth and color images and attempt to minimise a photometric reprojection error.

Our method is based on the RGBD-ICP algorithm of Henry *et al.* [4], which uses sparse visual keypoints to initialise ICP on dense point clouds. This combines the speed and robustness of keypoint matching with the accuracy of correctly initialised cloud alignment. However we use a different keypoint detector, Center Surround Extrema (CenSurE) [9], which is fast to compute and has demonstrated results in long-range outdoor stereo odometry [10]. We also employ a planar fit metric to determine whether to use RGBD or ICP odometry depending on terrain.

To constrain stereo odometry orientation errors, Konolige *et al.* [10] fuse odometry estimates with absolute roll and pitch and relative yaw measured by a navigation-grade inertial measurement unit. In a similar manner we employ

the AHRS of Mahony *et al.* [11] to compute an absolute orientation relative to a North-East-Down (NED) coordinate system using a low-cost IMMU. This orientation is fused with the pose estimate using a quaternion-based EKF.

III. RGBD IMMU Odometry

In this section we outline our RGBD and IMMU odometry system. The aim is to reconstruct the camera pose relative to the starting location in an NED coordinate frame. A schematic representation of the steps involved is shown in Fig. 2

A. Image Mapping

The Kinect v2 depth and color cameras are physically seperated and have different fields of view, resolutions and focal lengths. To find 3D keypoint locations the depth image D_d must be mapped into the color frame D_c. Fig. 3 illustrates the mapping process and camera parameters. First depth pixels with range value R_d at pixel coordinates (u_d, v_d) are projected to 3D coordinates in the depth camera frame (x_d, y_d, z_d). As shown in Eqn. 1 this requires the focal length of the depth camera in the x and y directions $(f_{x,d}, f_{y,d})$ and the centre pixel location $(c_{x,d}, c_{y,d})$.

$$(x_d, y_d, z_d) = \left(\frac{R_d(u - c_{x,d})}{f_{x,d}}, \frac{R_d(v - c_{y,d})}{f_{y,d}}, R_d \right) \quad (1)$$

Camera focal length, image center, and the relative pose between the Kinect color and depth cameras ${}^{D}T_{C}$ were found using Bouget's toolbox for stereo calibration [12]. This transform ${}^{D}T_{C}$ is used to map points in the depth camera frame to color frame coordinates (x_c, y_c, z_c) as per Eqn. 2.

$$\begin{bmatrix} x_c & y_c & z_c & 1 \end{bmatrix}^T = {}^{T}_{C} \begin{bmatrix} x_d & y_d & z_d & 1 \end{bmatrix}^T \quad (2)$$

Finally Eqn. 3 shows the reprojection to pixel coordinates in the color image, where $\lfloor x \rceil$ is the round-to-nearest-integer function on x, and the pixel (u_c, v_c) is assigned a range value $R_c = z_c$. Focal lengths and camera centres for the color camera follow the same subscript notation as for the depth camera. This mapping is implemented in parallel on the GPU using a single thread per depth image pixel.

$$(u_c, v_c) = \left(\left\lfloor \frac{x_c f_{x,c}}{d} - c_{x,c} \right\rceil, \left\lfloor \frac{y_c f_{y,c}}{d} - c_{y,c} \right\rceil \right) \quad (3)$$

B. Keypoint Detection

The Star keypoint detector is the OpenCV implementation of CenSurE. It looks for a region of light surrounded by a region of dark or vice versa at different sizes using a star shaped filter kernel shown in Fig. 4. The star is the addition of a square and a diamond shape which can be quickly computed at arbitrary sizes using rectangular and triangular integral images. The filter response is the difference between a larger star shape and a smaller one; specifically in Fig. 4 it is the sum of multiplying every pixel intensity in the light grey region by 1, the dark grey by 2, the white by -1 and the inner hatched area by -2.

Keypoints are selected by thresholding the response and supressing local non-maximums and keypoints on edges. In our implementation, the integral image and non-maxima suppression is processed on the CPU, while each GPU thread computes the response for a single pixel at each filter size.

C. Keypoint Description and Matching

The Binary Robust Independent Elementary Feature (BRIEF) descriptor is a fast metric for keypoint matching that describes a keypoint with a 256 bit binary string of random pixel pair intensity comparisons [13]. The similarity between two descriptors is their Hamming distance - the number of bits which are different, which can be efficiently computed using hardware based population count instructions on the exclusive OR of the two strings. As such we perform brute-force matching strategy on the GPU, where each GPU thread compares a BRIEF descriptor in one keypoint set to every descriptor in the other set.

D. Inlier Filtering

The set of keypoint matches is filtered to select good matches. First Lowe's test for rejecting visually ambiguous matches is applied [14]. This requires the Hamming distance between the descriptors for the best match to be better than the distance to the second-best match by some factor. We use a factor of 0.6.

The second filter, RANSAC [15], uses geometrical constraints to estimate a mutually consistent inlier set. The process is outlined in Algo. 1. For *maxIter* iterations, a random sample set s of 3 points is drawn from the N matches between the current and previous image keypoint sets, kp_{curr} and kp_{prev}. This set is used to estimate a transform T. Each keypoint k in the previous image at point (x_k, y_k, z_k) is mapped by T to a predicted point (x_p, y_p, z_p) and reprojected to predicted image coordinates (u_p, v_p). An error ε is the Euclidean distance between the predicted coordinates and the measured match location in the current frame $(u_{k,curr}, v_{k,curr})$. A threshold on this error determines whether the match k is an inlier. The maximal inlier set across all iterations is the output of RANSAC.

In our implementation the CPU selects the random sample and estimates a transform for one iteration, while simulatenously each GPU thread transforms, reprojects and computes the error for a single keypoint using the transform T from the previous iteration.

E. Keypoint Projection and Cloud Alignment

Once keypoints have been detected and matched, they are projected to a 3D point. As the color and depth cameras have different resolutions a keypoint k in the color image will usually not have a mapped range reading R_k at its pixel coordinates (u_k, v_k). A range must instead be interpolated from the mapped depth image, where $d(u, v)$ is the range at coordinates (u, v). The interpolation is performed over a 7x7 window around the keypoint pixel. A weighting $w_{i,j}$ at coordinates (i, j) relative to the keypoint location, given in

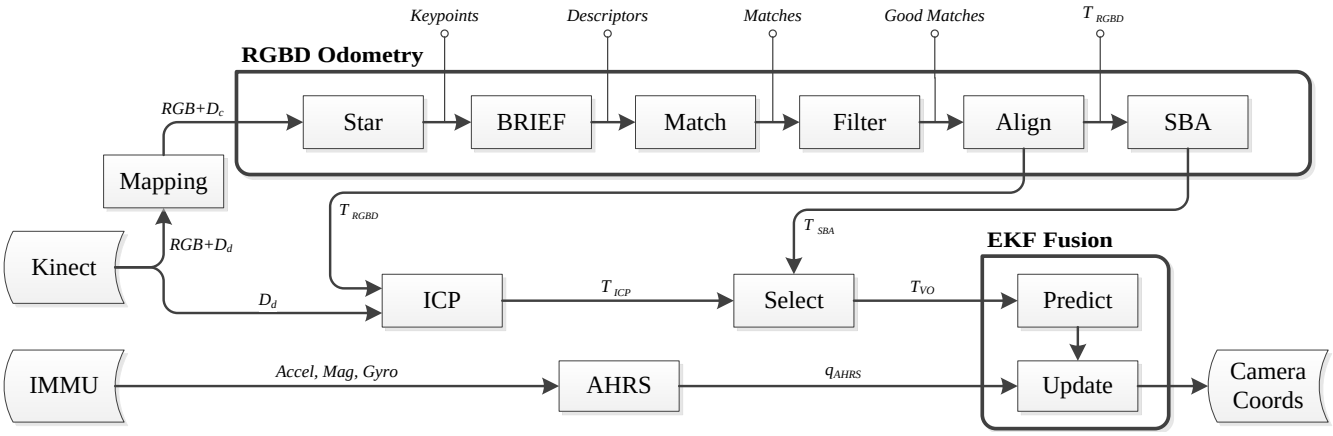

Fig. 2. Flowchart for the combined RGBD and IMMU odometry processing

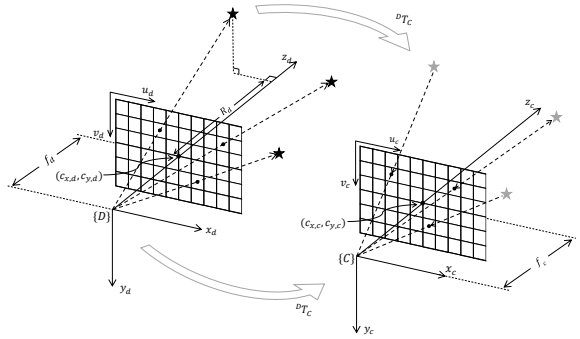

Fig. 3. Mapping from the Kinect depth to color frames

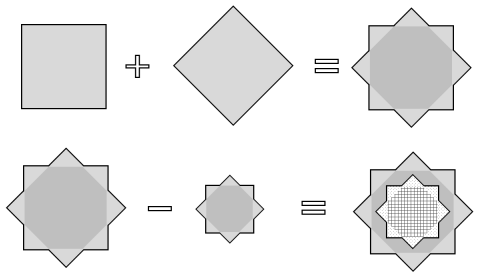

Fig. 4. The Star keypoint detector kernel

Eqn. 4, puts more weight on readings closer to the target location.

$$w_{i,j} = \begin{cases} \frac{1}{i^2+j^2} & if\ d(u_k+i, v_k+j) > 0 \\ 0 & otherwise,\ or\ if\ \ j=i=0 \end{cases} \quad (4)$$

The sum of weighted pixel values is normalized by the total weightings to give the range estimate, as per Eqn. 5. Projection to a 3D point is then be performed using Eqn. 1 albeit with the equivalent color camera parameters.

$$R_k = \Sigma_{i=-3}^{3}\Sigma_{j=-3}^{3}\left(\frac{w_{i,j}\times d(u_k+i, v_k+j)}{\Sigma_{i=-3}^{3}\Sigma_{j=-3}^{3}(w_{i,j})}\right) \quad (5)$$

To estimate the camera pose the matched keypoint clouds

Algorithm 1 RANSAC inlier finding procedure

```
maxInliers ← Inliers ← 0
for i < maxiters do
    s ← GetRandSample(N,3)
    T ← EstimateTransform(kp_curr(s), kp_prev(s))
    for each keypoint k in kp_prev do
        (x_p, y_p, z_p) ← TransformPt(x_k, y_k, z_k, T)
        (u_p, v_p) ← Reproject(x_p, y_p, z_p)
        ε ← (u_p − u_k,curr)^2 + (v_p − v_k,curr)^2
        if ε < thresh then
            Add k to Inliers
        end if
    end for
    if size(Inliers) > size(maxInliers) then
        maxInliers ← Inliers
    end if
end for
```

a and b are aligned using Umeyama's method [16]. The transform between them $T_{RGBD} = [R,t]$ comprises the rotation matrix R and translation t, and is found by minimising a Euclidean distance point to point error metric ε in Eqn. 6. Umeyama's method is an efficient closed form solution to this minimization problem using the singular value decomposition of the covariance between the point clouds.

$$\varepsilon = \Sigma_{i=1}^{n}||Ra_i + t - b_i||^2 \quad (6)$$

F. Sparse Bundle Adjustment

In order to minimize drift and accumulated errors, detected keypoints are tracked across multiple frames for use in a bundle adjustment process. For n camera poses T_{SBA} and the set of m observed 3D points P then an error function ε is defined in Eqn. 7 as the sum of the Euclidean distances between predicted and measured keypoint locations. Predicted locations are found by transforming a world point P_j into local camera coordinates by applying transform $T_{SBA,i}$ and reprojecting back to image coordinates, while (u_{ij}, v_{ij}) is the measured keypoint location of point j in image frame i.

$$\varepsilon = \Sigma_{i=1}^{n}\Sigma_{j=1}^{m}||Reproj(P_j, T_{SBA,i}) - (u_{ij}, v_{ij})||^2 \quad (7)$$

The SBA library [17] efficently solves this minimization problem for both camera poses T_{SBA} and keypoint locations P by recognising that the Jacobian is sparse, containing mostly zero elements. In our system SBA is applied to keypoints tracked across the last ten frames.

G. Dense ICP

The iterative closest point algorithm aligns two point clouds without any prior knowledge of point correspondances [7]. The original implementation of finding closest points is prohibitively slow for dense point clouds, instead KinectFusion uses projective data assocation [6]. The transform $T_{ICP} = [R|t]$ of rotation matrix R and translation t between the clouds minimizes the error metric ε which is the distance between a point b_i in one cloud and a tangent plane in the other defined by the point a_i and the normal at that point n_i. This has been shown to converge faster [18].

$$\varepsilon = \Sigma_i((Ra_i + t - b_i) \cdot n_i)^2 \quad (8)$$

This is a nonlinear function, however a closed form minimum solution is obtained by making a linearising small-angle approximation to the rotation matrix as described in [18]. By initialising with T_{RGBD}, the small angle assumption is more likely to hold and ICP is more likely to iterate to the correct local minimum.

H. Selecting the Odometry Source

With two different odometry modalities available, we select the one more likely to give a better result as the odometry output T_{VO} using a switching strategy. ICP is preferred, as when T_{RGBD} is accurate then ICP should confirm the same pose, and when T_{RGBD} is less accurate due to outliers ICP should iterate towards the correct pose change. However this is only true when there is some depth variation to constrain the point cloud alignment. On near planar surfaces like short grass ICP is likely to settle into an incorrect local minimum.

To determine which estimate to use, the depth image is reduced to an 8x8 image by averaging and projected to a 64 point cloud using a virtual camera with focal length $f = 1$. Only if the residuals of a planar fit to this cloud are too small then the surface is nearly flat and T_{SBA} is selected over T_{ICP}.

I. AHRS and Orientation Fusion

The AHRS used is an open-source implementation of the work of Mahoney *et al.* [11] which computes a rotation matrix describing the orientation of the IMMU relative to an NED coordinate system. To simplify later calculations, the output of the AHRS is precomputed to a quaternion using the method in [19]. Applying the relative rotation between the camera and IMMU, found using the toolbox of Lobo and Dias [20], gives the orienation of the camera in the NED frame.

The selected incremental pose estimates T_{VO} are then fused using an EKF with the orientation reading q_{AHRS} provided by the AHRS. The basic equations of the EKF can be found in [21]. The state vector $\mathbf{x} = [q,t]$ to estimate comprises a quaternion orientation $q = (q_w, q_x, q_y, q_z)$ and a translation $t = (x,y,z)$. The *a priori* state prediction $\hat{\mathbf{x}}_k^-$ at time k is given by the dynamic model f in Eqn. 9, where the previous state $\hat{\mathbf{x}}_{k-1}$ is compounded with the odometry motion estimate $\mathbf{u}_{k-1} = \Omega(T_{VO})$ where Ω maps T_{VO} to a quaternion q_u and translation t_u. Here $\otimes$ is the quaternion multiplication operator.

$$\hat{\mathbf{x}}_k^- = f(\hat{\mathbf{x}}_{k-1}, \mathbf{u}_{k-1}) = \begin{bmatrix} q_{k-1} \otimes q_u \\ t_{k-1} + q_{k-1} \otimes t_u \otimes q_{k-1}^{-1} \end{bmatrix} \quad (9)$$

The EKF requires the Jacobian $\mathbf{A}$ of the prediction function in order to propagate the state covariance estimate $\mathbf{P}$. Eqn. 10 gives the exact form of $\mathbf{A}$.

$$\mathbf{A} = \frac{df}{d\mathbf{x}} = \begin{bmatrix} q_w & -q_x & -q_y & -q_z & \\ q_x & q_w & q_z & -q_y & \mathbf{0}_{4,3} \\ q_y & -q_z & q_w & q_x & \\ q_z & q_y & -q_x & q_w & \\ & \mathbf{0}_{3,4} & & & \mathbf{I}_{3,3} \end{bmatrix} \quad (10)$$

By precomputing the AHRS orientation to a quaternion the matrix $\mathbf{H}$ which maps predicted state $\hat{\mathbf{x}}_{k-1}^-$ to predicted measurement $\hat{z}_k$ simply selects the quaternion part of the state vector, $\mathbf{H} = \begin{bmatrix}\mathbf{I}_{4,4} & \mathbf{0}_{4,3}\end{bmatrix}$.

The AHRS measurements are asynchronous to the camera frames, so the measured orientation z_k is spherically interpolated using the method in [19] between the AHRS readings before and after the current frame timestamp. Standard Kalman Filter equations then apply to obtain the updated *posteriori* state vector $\hat{\mathbf{x}}_k^+$ and convarience estimates $\hat{\mathbf{P}}_k^+$.

The system and measurement noise covariance matrices, $\mathbf{Q}$ and $\mathbf{R}$ respectively, were set empirically in proportion to the noise of the odometry prediction and AHRS update subsystems when stationary. A covariance noise of 0.0001 was used for the odometry translation estimate and AHRS orientation measurement, while 0.001 was used for the odometry quaternion rotation component due to its cumulative drift.

IV. Experimental Results

In our experiments RGBD data is captured using a Kinect for Windows v2 camera, and the IMMU is the low-cost Sparkfun Razor IMU. The Kinect measures depth using a pulsed IR time of flight camera, which can be corrupted by sunlight. Our setup, shown Fig. 5, shields the camera field of view with a translucent frustum shape. This blocks enough IR light for the depth camera to operate normally, while transmitting sufficient visible light for the color camera. The IMMU is rigidly attached but seperated from the camera to minimize EM interferance, as shown in Fig. 6. A GPS unit also collects a ground truth reference.

Two sets of odometry results are presented. The first was conducted at midday, using the frustum to shield the Kinect. The path was flat and the Kinect ground facing. The second was conducted in late afternoon shadow with the Kinect camera hand-held while walking along a rough bush track. We also present timings for GPU implementations of several odometry processing steps.

Fig. 5. The frustum which shields the Kinect field of view from sunlight

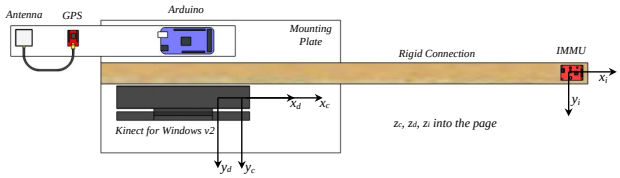

Fig. 6. Kinect and IMMU configuration

A. Ground Facing Camera with Frustum

The ground facing experiment using the frustum was conducted at Tunks Park oval, on a terrain of short grass at speeds of around 60cm/sec. This means little depth information was available and the SBA odometry was chosen at every step over ICP. Fig. 7 shows the side view of the odometry estimate from SBA alone. While the GPS path is flat, compounding rotational errors cause the odometry estimate to spiral into the sky. Upon fusing with the AHRS, the path is correctly constrained to a flat plane as shown in Fig. 8.

However, errors in AHRS compass orientation still cause unsatisfactory results as the metal frame supporting the frustum distorts the local magnetic field. To compensate we created a lookup table of the magnetic deviation due to the frustum at 10° increments in bearing, which allows the odometry estimate to closely track the GPS path.

The straight line distances of the initial backwards E-shape were physically measured using a surveyors wheel. The actual distances and odometry estimates for the first six legs are shown in Tab. I, along with the percentage error and the drift normalized to the time taken for each leg as a drift per second. Drifts of less than 2.5cm/sec were achieved on each leg, while at the most Western point of the path, furthest from the start, the position error is 6.5m, or 1.2% of the 550m travelled.

B. Hand Held Camera Bushwalk

The second trial was conducted along a bush trail at Harold Reid park in the afternoon shade. Terrain included small trees and plants, leaf litter, stone steps, gulleys and gravel. This presents a challenging dataset to both sensing types. The low light and hand-held motion make motion blur a

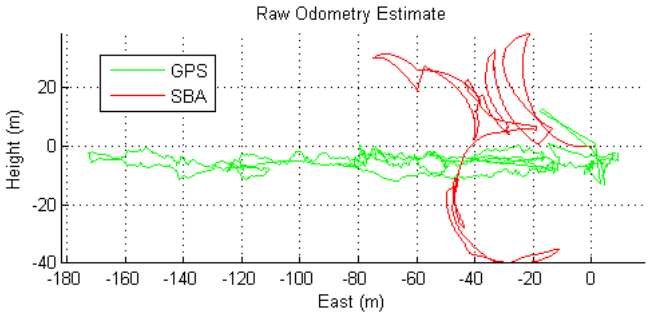

Fig. 7. Cumulative orienation drift from SBA

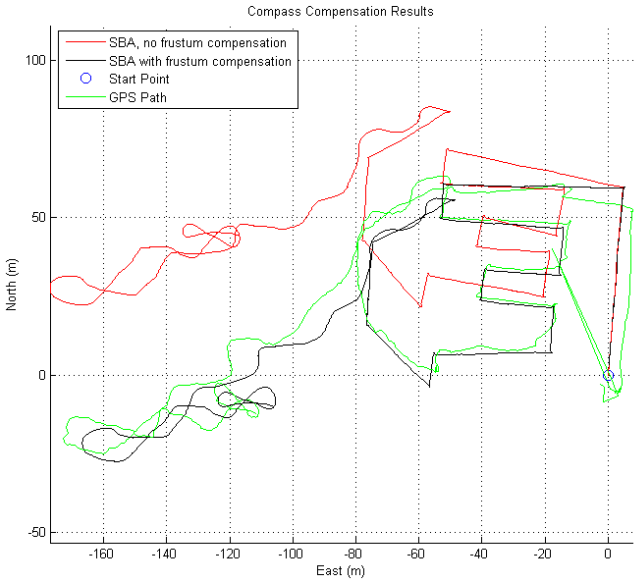

Fig. 8. Ground facing odometry estimate fused with AHRS

significant issue for visual keypoint detection. There are also some sections where the path opens to nearly flat providing few features for localization using ICP.

Results from this experiment are shown in Fig. 9. T_{ICP} was primarily selected, however the path from T_{SBA} is shown for comparison purposes - it fails to maintain a good estimate when there are few inlier matches due to motion blur. By contrast, despite the additional noise in the GPS position descending into the valley, the ICP odometry clearly tracks the GPS path. At the end of the 600m path, the error in position is 20.6m, or 3.4% of the distance travelled to that point.

C. Timing

The timings for a serial and parallel implementation for relevant odometry processing steps, with the corresponding improvement factor in speed, are given in Tab. II. The input data are 1920x1080 color images and 512x424 dense depth images, with around 5000 keypoints found per image. These timings are presented using an Intel i5-3570 processor and an NVidia GTX580 graphics card running the Compute Unified Device Architecture (CUDA).

TABLE I

LINEAR DISTANCE ACCURACY FOR GROUND FACING PATH

Leg	Actual (m)	Est. (m)	Err. (%)	Drift (m/s)
1	60.2	59.3	1.49	0.0085
2	59.7	56.9	4.69	0.0250
3	10.8	10.6	1.95	0.0095
4	39.9	38.6	3.25	0.0173
4	15.0	14.5	3.33	0.0156
5	24.9	23.8	4.42	0.0229

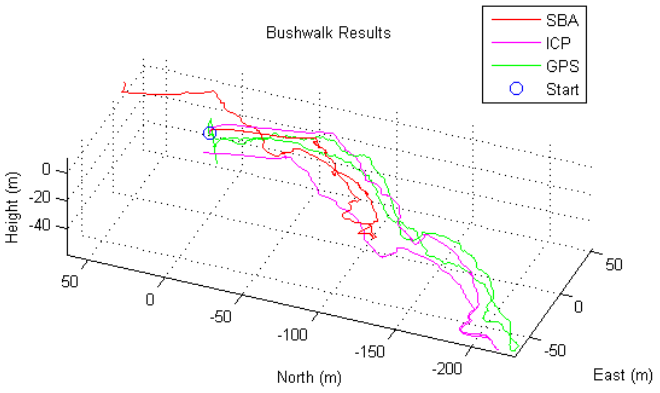

Fig. 9. Bushwalk odometry estimate fused with AHRS

A significant increase in speed is attained for mapping the depth image by parallel processing. However computation of the Star detector response suffers from a bottleneck as the integral image computation and non-maxima suppression are performed by the CPU. The BRIEF descriptor and brute-force matching steps both require a large number of accesses to the slower global CUDA memory, which limit the speed increase. RANSAC also experiences only modest gains, as the CPU is doing the work of selecting the sample and finding the minimising transform for each iteration, providing another bottleneck. However overall these parallelizations greatly reduce the frame-to-frame processing time.

TABLE II

PARALLELIZATION TIMING

Step	Serial (s)	Parallel (s)	Improvement Factor
Mapping	0.3455	0.0054	63.68
Star	0.2275	0.0904	2.52
BRIEF	0.4464	0.0197	22.70
Matching	0.2409	0.0150	16.07
RANSAC	1.0189	0.1770	5.77

V. CONCLUSION

In this paper we described a system for outdoor odometry estimation using RGBD cameras and presented an experimental setup enabling outdoor use. We employed both keypoint matching and ICP localization, and effectively switched between them depending on the terrain. EKF fusion with AHRS readings from an IMMU were used to constrain the orienation estimate. Final position errors of less than 4% and drifts of less than 25mm/sec over hundreds of meters were achieved in a variety of terrain and motion conditions, and GPU implementations of processing steps achieved speed increases by a factor of between 2-60.

REFERENCES

[1] J. Leonard and H. Durrant-Whyte, "Mobile robot localization by tracking geometric beacons," *Robotics and Automation, IEEE Transactions on*, vol. 7, no. 3, pp. 376–382, Jun 1991.

[2] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry," in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. of the 2004 IEEE Computer Society Conference on*, vol. 1. IEEE, 2004, pp. 652–659.

[3] B. Barshan and H. F. Durrant-Whyte, "Inertial navigation systems for mobile robots," *Robotics and Automation, IEEE Transactions on*, vol. 11, no. 3, pp. 328–342, 1995.

[4] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "Rgb-d mapping: Using depth cameras for dense 3d modeling of indoor environments," in *Experimental Robotics*. Springer, 2014, Conference Proceedings, pp. 477–491.

[5] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy, "Visual odometry and mapping for autonomous flight using an rgb-d camera," in *International Symposium on Robotics Research (ISRR)*, 2011.

[6] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, *et al.*, "Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera," in *Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM, 2011, pp. 559–568.

[7] P. J. Besl and N. D. McKay, "A method for registration of 3-d shapes," in *Robotics-DL tentative*. International Society for Optics and Photonics, 1992, pp. 586–606.

[8] C. Kerl, J. Sturm, and D. Cremers, "Robust odometry estimation for RGB-D cameras," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 3748–3754.

[9] M. Agrawal, K. Konolige, and M. R. Blas, "CenSurE: Center surround extremas for realtime feature detection and matching," in *Computer Vision–ECCV 2008*. Springer, 2008, pp. 102–115.

[10] K. Konolige, M. Agrawal, and J. Sola, "Large-scale visual odometry for rough terrain," in *Robotics Research*. Springer, 2011, pp. 201–212.

[11] R. Mahony, T. Hamel, and J.-M. Pflimlin, "Nonlinear complementary filters on the special orthogonal group," *Automatic Control, IEEE Transactions on*, vol. 53, no. 5, pp. 1203–1218, 2008.

[12] J.-Y. Bouguet. (2004) Camera calibration toolbox for matlab. [Online]. Available: http://www.vision.caltech.edu/bouguetj/calib_doc/

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

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

[15] 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.

[16] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 13, no. 4, pp. 376–380, 1991.

[17] M. I. Lourakis and A. A. Argyros, "SBA: A software package for generic sparse bundle adjustment," *ACM Transactions on Mathematical Software (TOMS)*, vol. 36, no. 1, p. 2, 2009.

[18] K.-L. Low, "Linear Least-Squares Optimization for Point-to-Plane ICP Surface Registration," University of North Carolina, Report Tech. Rep. TR04-004, 2004.

[19] E. B. Dam, M. Kock, and M. Lillholm, "Quaternions, interpolation and animation," University of Copenhagen, Report Tech. Rep. DIKU-TR98/5, 1998.

[20] J. Lobo and J. Dias, "Relative pose calibration between visual and inertial sensors," *The International Journal of Robotics Research*, vol. 26, no. 6, pp. 561–575, 2007.

[21] G. Welch and G. Bishop, "An introduction to the kalman filter," University of North Carolina - Department of Computer Science, Tech. Rep. TR 95-041, July 2006.