# Scene Reconstruction from Uncontrolled Motion using a Low Cost 3D Sensor

Pierre Joubert and Willie Brink*
Applied Mathematics
Department of Mathematical Sciences
University of Stellenbosch, South Africa
*Email: wbrink@sun.ac.za

***Abstract*—The Microsoft Kinect sensor provides real-time colour and depth data via a colour and infrared camera. We present an approach that uses this inexpensive device for 3D reconstruction from uncontrolled motion. We obtain the intrinsic and extrinsic parameters of a Kinect by calibration, and find a mapping from the raw depth data to real world metric coordinates empirically. With these parameters we are able to create a dense 3D point cloud reconstruction, with colour data, of a scene recorded in a single frame by the Kinect. We then adapt a feature-based approach to align point clouds reconstructed from different, uncontrolled points of view of the same static scene. We demonstrate the effectiveness of this approach by means of examples, and find that success hinges upon the availability and accuracy of feature matches across the different views.**

## I. Introduction

The Microsoft Kinect, released in November 2010, is a consumer grade 3D sensor, capable of giving as output 11-bit depth data in real-time, at 30 frames per second, and at a resolution of $640 \times 480$ [1]. This sensor, shown in Fig. 1, was originally developed to serve as a controller-free input device for the Xbox 360 but has quickly gained popularity in the computer vision research community.

The Kinect estimates distance by projecting a fixed infrared (IR) pattern onto nearby objects which a dedicated IR camera picks up. The distortion of the pattern, due to the structure of the scene, reveals depth.

Some applications that may benefit immensely from such a low cost 3D sensor include the estimation of body position, gestures and motion for human-computer interaction [2]; map generation for robotic navigation [3]; obstacle detection and avoidance [4]; and general 3D object reconstruction.

The Kinect also provides 8-bit colour (RGB) data simultaneously with depth data, at the same rate and resolution. Fig. 2 gives an example. However, it is important to note that the depth and RGB data are given relative to the IR camera and RGB camera respectively. This makes simultaneous use of depth and RGB data non-trivial. Fig. 2 indicates this well, where it is clear that the IR image is more "zoomed-in" than the RGB image.

## II. Problem statement and approach

The objective of this paper is to firstly render a 3D cloud of points, with colours, from the depth and RGB data captured by the Kinect, in metric space. After achieving this we want to align point clouds from consecutive frames (each from a slightly different angle) to form a dense 3D reconstruction of a static scene.

We will approach these objectives by dividing the tasks into two main parts, namely the *sensor calibration* step and the *application* step.

### Sensor calibration

- The IR and RGB cameras need to be calibrated separately, in order to find intrinsic parameters such as focal lengths, centres of projection and radial distortion coefficients for each. This will facilitate the mapping of points (or pixels) between image and real world coordinates.
- Stereo calibration must be performed on the two cameras (IR and RGB), to find a relative rotation and translation from one to the other. This will enable the simultaneous use of depth and colour information.
- A mapping from raw depth data to real world metric coordinates needs to be determined, which will aid in the *application* step.

### Application

- After calibration, pixels in a captured depth image can be mapped to real world metric coordinates. This will produce a metric point cloud.
- The point cloud can be projected onto the RGB image plane, so that each point is assigned a colour value. This will result in the desired colour point cloud.
- Corresponding features in successive RGB images, taken of a static scene from different viewpoints, can be matched to find a relative rotation and translation from one image to the other. This will assist in point cloud alignment which in turn may result in a dense 3D reconstruction.

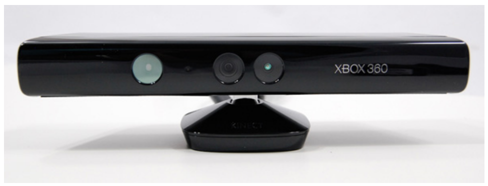

Fig. 1. The Microsoft Kinect sensor with IR projector, RGB camera and IR camera visible from left to right.

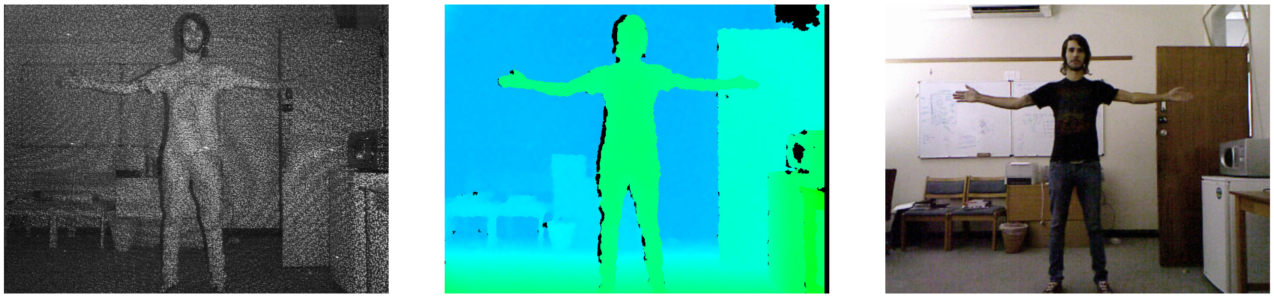

Fig. 2. Example IR, depth and RGB output images from the Kinect. Depth is calculated onboard from the distortion of the IR pattern and is given relative to the position of the IR camera. The RGB camera is completely separate and the sensor needs to be calibrated for colour and depth to be used simultaneously.

## III. Sensor calibration

This section describes our approach for calibrating a Kinect. It is important to state that the intrinsic and extrinsic parameters we obtain may differ slightly from those of another Kinect, due to small mechanical inaccuracies in the manufacturing process.

We first consider single camera calibration, in order to find the intrinsic parameters of the IR and RGB cameras separately, and then stereo calibration to find the position and orientation of the RGB camera relative to the IR camera. Finally we discuss our approach for inferring a mapping from raw depth values (as given by the Kinect) to metric 3D coordinates.

### *A. Single camera calibration*

Under the pinhole camera model [5], a point $\mathbf{X}$ in some fixed world coordinate system is mapped to the image plane of an arbitrary camera by the equation

$$\mathbf{x} = \mathbf{KR}\big[\mathbf{I} \mid -\mathbf{c}\big]\mathbf{X} = \mathbf{PX}. \tag{1}$$

Here $\mathbf{X}$ is a 4-element homogeneous vector measured in the world coordinate system, and $\mathbf{x}$ a 3-element homogeneous vector measured on the image plane from which pixel coordinates are obtained. The $3 \times 3$ matrix $\mathbf{K}$ is called the calibration matrix and contains intrinsic parameters such as focal length, principal point offset and skewness factor. The $3 \times 3$ rotation matrix $\mathbf{R}$ and the inhomogeneous 3-element translation vector $\mathbf{c}$ provide the extrinsic parameters of the camera which relate the camera's coordinate system with the world. The $3 \times 4$ matrix $\mathbf{P} = \mathbf{KR}\big[\mathbf{I} \mid -\mathbf{c}\big]$ is called the camera matrix.

The calibration matrix is of the form

$$\mathbf{K} = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2}$$

where $\alpha_x$ and $\alpha_y$ are the focal lengths in pixels, $x_0$ and $y_0$ the coordinates of the image centre in pixels, and $s$ the skew factor [5]. These 5 intrinsic parameters describe the inner workings of the camera.

A standard way of obtaining the intrinsic parameters is to capture a known object and observe how it distorts under the projective mapping introduced by the camera (which is essentially the multiplication by $\mathbf{P}$). A planar checkerboard, with known measurements, is often used as calibration object as corners are easily identifiable in the image and can be found automatically to a high degree of accuracy. Each of the imaged corner positions, with its corresponding known world coordinates (measured on the checkerboard plane), is substituted into (1). This leads to a linear system from which $\mathbf{P}$ can be solved. We typically use many images of the checkerboard held at various positions and orientations, to solve for $\mathbf{P}$ robustly in a least-squares sense.

Real cameras, however, rarely adhere to the idealized pinhole model. The lens may distort the image in such a way that straight lines in the world are not straight in the image. This nonlinear distortion is usually radial, where regions furthest from the image centre are distorted most. It can be modelled mathematically, coefficients in a Taylor approximation of this model can be determined, and the image can thence be undistorted.

Camera calibration is therefore usually an iterative process: the linear intrinsic parameters are found, an estimate for the lens distortion coefficients are obtained and images are undistorted, the linear parameters are re-estimated from these new images, the distortion coefficients are updated, and so on until convergence.

In calibrating the IR and RGB cameras of a Kinect we place a planar checkerboard in view of both, at various positions and angles, and capture images. Examples are shown in Fig. 3. The images are fed to Bouguet's camera calibration toolbox [6], [7] which performs iterative optimization in much the same manner as the process described above.

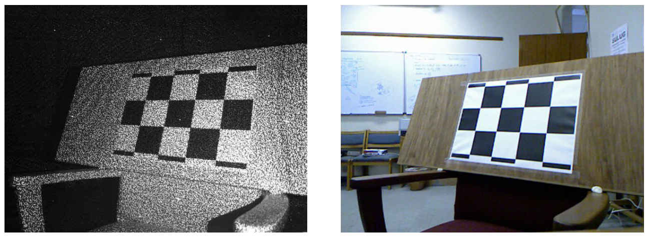

Fig. 3. An example of the planar checkerboard pattern used for calibration, as seen by the IR (left) and RGB (right) cameras. These images are used in calibrating each camera separately, as well as in stereo calibration where they act as an image pair.

It should be noted that the IR images will always be quite heavily "corrupted" by the infrared pattern (used to infer depth) which may cause problems for the corner extractor. A filter such as the adaptive median filter [8] can be effective here. In fact, the corners are still fairly salient in the raw IR images and we can find them quite accurately even in the presence of noise.

After calibrating the IR and RGB cameras separately we obtained the following calibration matrices:

$$\mathbf{K}_{\text{ir}} = \begin{bmatrix} 593.73 & 0 & 315.19 \\ 0 & 591.72 & 219.72 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3}$$

$$\mathbf{K}_{\text{rgb}} = \begin{bmatrix} 522.97 & 0 & 335.67 \\ 0 & 521.16 & 243.42 \\ 0 & 0 & 1 \end{bmatrix}, \tag{4}$$

rounded for display purposes. Distortion coefficients were also calculated for each camera.

## *B. Stereo calibration*

Stereo calibration is necessary when a system consists of two cameras, and finds the rotation and translation from one camera to the other (i.e. the extrinsic parameters). It follows the same methodology as single camera calibration, except that two images of the calibration object are captured simultaneously by the two cameras. We then know that imaged features (or corners) of the object will have the same world coordinates in the two images. This enables the calculation of a spatial relationship between the two cameras.

Bouguet's toolbox [6] can be used here as well, on a set of image pairs, to determine the rotation **R** and translation **c** from one camera (in our case the IR camera) to the other (the RGB camera). For our Kinect we obtained

$$\mathbf{R} = \begin{bmatrix} 0.99 & 0.01 & -0.05 \\ -0.01 & 0.99 & 0.00 \\ 0.05 & 0.00 & 0.99 \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} -23.01 \\ 3.14 \\ 1.74 \end{bmatrix}, \tag{5}$$

rounded to two decimals for display purposes. Here the translation is given in millimetres. These parameters suggest that the two cameras are horizontally about 23mm apart and almost perfectly aligned ($\mathbf{R} \approx \mathbf{I}$). This agrees with a visual inspection of the Kinect.

The results in (3), (4) and (5) allow us to write down the two camera matrices as

$$\mathbf{P}_{\text{ir}} = \mathbf{K}_{\text{ir}} \left[ \mathbf{I} \mid \mathbf{0} \right], \quad \mathbf{P}_{\text{rgb}} = \mathbf{K}_{\text{rgb}} \, \mathbf{R} \left[ \mathbf{I} \mid -\mathbf{c} \right]. \tag{6}$$

These matrices define a relationship between depth data and colour data. A remaining issue is that of finding a general mapping from raw depth data (as returned by the Kinect in the form of range images) to real world metric coordinates.

## *C. Mapping raw depth to metric coordinates*

The Kinect gives as output 11-bit depth images, each of which being a $480 \times 640$ matrix of values between 0 and 2047 (an example is visualized in Fig. 2). We wish to transform each of these values to a point in 3D Euclidean space.

It turns out that the value of a particular pixel in the depth image indicates (in some way) the distance between the principal plane of the IR camera and a point in the world, measured perpendicularly to the principal plane through that pixel [1]. Therefore, if the world coordinate system coincides with the coordinate system of the IR camera, as it does in (6), that distance would be the $Z$-coordinate of the point. By these arguments we should be able to establish a mapping of the form $Z = f(d)$, where $d$ is a value in the depth image.

In an effort to determine the mapping $f$ we conducted an experiment by moving a planar surface orthogonally to the IR camera's principal axis at known distances. Depth images were recorded for distances from $0.5\,\text{m}$ to $3.25\,\text{m}$ (which is more-or-less the effective range of a Kinect), in $0.25\,\text{m}$ increments. It should be mentioned that it is difficult to move a planar surface completely orthogonally to the principal axis but, since we have access to depth values, the accuracy of this motion can be measured and fine-tuned in real-time.

The experiment resulted in a depth measurement (retrieved from the appropriate depth image) associated with every known distance. A plot of this data, as in Fig. 4, makes it clear that there may be a linear relationship between the measured depth values and the inverse of actual depth.

This apparent relationship allows us to infer a general mapping, using linear least-squares on the depth values and the inverse of actual distances, as

$$Z = (3.1055 - 0.0028409\, d)^{-1}, \tag{7}$$

where the coefficients are rounded only for display.

$X$- and $Y$-coordinates for a pixel $(u, v)$ in the depth image can be determined straightforwardly by using (2) and (6) to write $\mathbf{P}_{\text{ir}} \begin{bmatrix} X & Y & Z & 1 \end{bmatrix}^T$, i.e. the mapping of point $(X, Y, Z)$ onto the image plane of the IR camera, as

$$\begin{bmatrix} \alpha_x & 0 & x_0 & 0 \\ 0 & \alpha_y & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_x X + x_0 Z \\ \alpha_y Y + y_0 Z \\ Z \end{bmatrix}, \tag{8}$$

where $\alpha_x$, $\alpha_y$, $x_0$ and $y_0$ are the intrinsic parameters of the IR camera (from (3) we assume that $s = 0$). In order to obtain

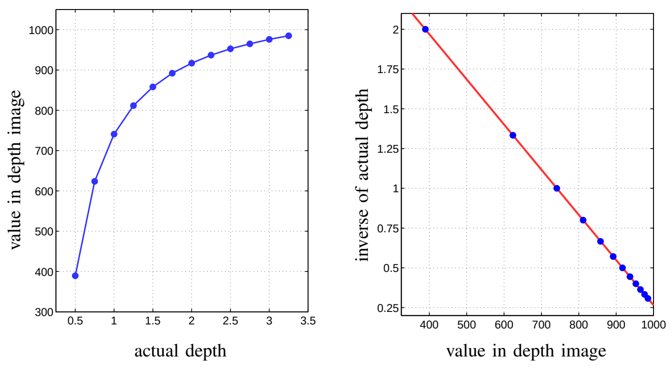

Fig. 4. Depth image data against known distances in metres (left) and the inverse of known distances against depth data with a least-squares line (right).

the 2D Euclidean version of this vector, which will be the IR image coordinates $(u, v)$, we divide by the third element so that

$$u = \alpha_x X/Z + x_0, \quad v = \alpha_y Y/Z + y_0. \tag{9}$$

Rearranging the above yields

$$X = (u - x_0)Z/\alpha_x, \tag{10}$$

$$Y = (v - y_0)Z/\alpha_y. \tag{11}$$

Equations (7), (10) and (11) can be used to map any pixel $(u, v)$ with value $d$ in the depth image to a point $(X, Y, Z)$ in Euclidean (real world) coordinates, in our case measured in millimetres. Fig. 5 illustrates with an example.

## IV. Application

The previous section dealt with the calibration of a Kinect. We proceed to explain how colours from the RGB image can be assigned to the 3D point cloud obtained and how various point clouds, reconstructed from different views of the same static scene, can be aligned.

### *A. Assigning colours to the point cloud*

Every point in the reconstructed point cloud has homogeneous coordinates of the form $\mathbf{X} = \begin{bmatrix} X & Y & Z & 1 \end{bmatrix}^T$, measured in the IR camera coordinate system. Since the RGB camera is calibrated with respect to the IR camera we can project such a point to the RGB image plane by calculating

$$\mathbf{x} = \mathbf{P}_{\text{rgb}}\mathbf{X}. \tag{12}$$

Note that radial lens distortion mentioned earlier has to be removed before this projection is performed.

We de-homogenize the vector $\mathbf{x}$ to arrive at pixel coordinates $(u, v)$ in the RGB image. These coordinates may very well be non-integer, necessitating some form of interpolation (such as nearest-neighbour, bi-linear or bi-cubic). We end up with a colour value which can be assigned to the point $\mathbf{X}$. Fig. 6 shows the result of this colour mapping on the point cloud in Fig. 5 (bi-linear interpolation was used).

### *B. Point cloud alignment*

Next we consider the problem of aligning a number of point cloud reconstructions of a scene, each taken from a different vantage point. The viewpoints are uncontrolled but we will assume that consecutive ones do not differ greatly, for sufficient overlap to exist.

Suppose we capture $n$ views of some static scene with the Kinect. Every one of these captured views consists of an IR image, an RGB image and a depth image. We map the pixels in the depth image to $(X, Y, Z)$ coordinates, project them to the RGB image and obtain a colour point cloud. These point clouds will be given relative to the IR camera and if we can estimate the motion of the IR camera from one point cloud to the next, we should be able to transform all the point clouds to a single fixed coordinate system, say the coordinate system of the IR camera corresponding to the first view.

The method discussed here is incremental, in the sense that a new point cloud is aligned with the one preceding it. Every new alignment will update a total rotation and translation, for transforming the cloud to the fixed coordinate system.

We exploit the assumption of sufficient overlap, and estimate motion of the RGB camera by means of a feature-based approach. It will therefore be convenient to transform the coordinates in a point cloud to the RGB camera's coordinate system. This is achieved easily:

$$\begin{bmatrix} X_{\text{rgb}} & Y_{\text{rgb}} & Z_{\text{rgb}} \end{bmatrix}^T = \mathbf{R}^T \begin{bmatrix} X & Y & Z \end{bmatrix}^T + \mathbf{c}, \tag{13}$$

with $\mathbf{R}$ and $\mathbf{c}$ given in (5).

Features in an image are salient, easily identifiable points and they typically have associated descriptors (depending on the feature detection method used). Ideally a descriptor should be scale, rotational and affine invariant. For this reason we employ the tried-and-tested Scale Invariant Feature Transform (SIFT) [9] to detect and match features across consecutive RGB images. A robust RANSAC estimator [10] can be used to remove outliers (incorrect matches) and we can then determine camera motion [5]. Fig. 7 gives an example.

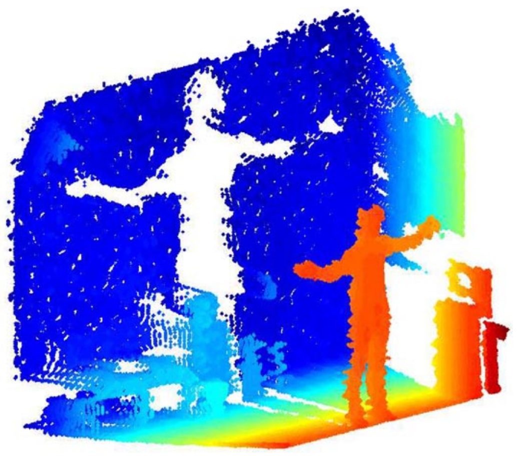
Fig. 5. A depth image mapped to a point cloud in Euclidean coordinates. Here points are coloured in the jet colour scheme according to depth.

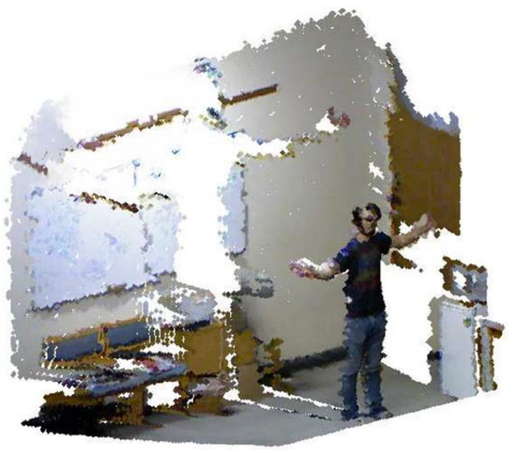
Fig. 6. Every point from the point cloud shown in Fig. 5 is projected to the RGB image (shown in Fig. 2), and colour is assigned thusly.

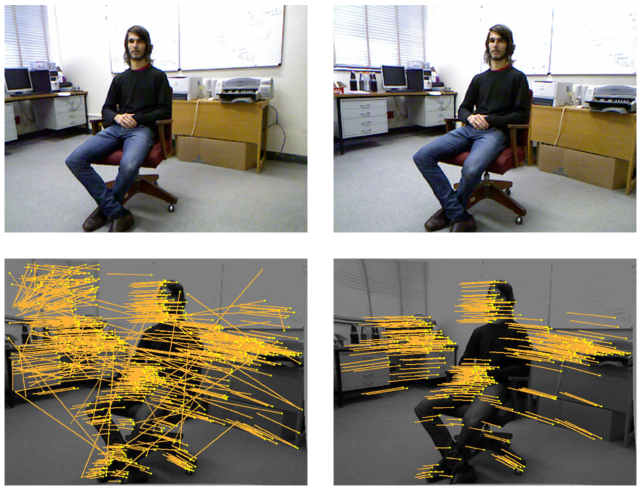

Fig. 7. Matching SIFT features determined for the two images shown on top. Putative matches are shown in the bottom left, and inliers remaining after RANSAC-based motion estimation in the bottom right.

Suppose, for a pair of RGB images, the estimated motion that takes the second camera's coordinate system to the first is represented by a rotation matrix $\mathbf{R}_m$ and translation vector $\mathbf{c}_m$. It is important to note that $\mathbf{c}_m$ is retrievable only up to scale. However we can determine this scale, at least in theory, by using the known metric 3D coordinates of matched features. These coordinates are at our disposal from the point clouds constructed by (7), (10), (11) and (12) for each image.

Consider homogeneous image coordinates $\mathbf{x}_1$ and $\mathbf{x}_2$ that were found to be a feature correspondence in the two RGB images. Furthermore let $\tilde{\mathbf{X}}_1$ be the inhomogeneous 3D coordinates of $\mathbf{x}_1$ in the first camera's coordinate system, and $\tilde{\mathbf{X}}_2$ the inhomogeneous 3D coordinates of $\mathbf{x}_2$ in the second camera's coordinate system. We therefore have

$$\tilde{\mathbf{X}}_1 = \mathbf{R}_m^T \tilde{\mathbf{X}}_2 + \lambda \mathbf{c}_m, \tag{14}$$

with $\lambda$ the unknown scale. Rewriting this expression as

$$\lambda \mathbf{c}_m = \tilde{\mathbf{X}}_1 - \mathbf{R}_m^T \tilde{\mathbf{X}}_2 \tag{15}$$

provides three equations (per feature match) from which $\lambda$ can be solved. Due to noise in the depth data and slight inaccuracies in the estimated motion parameters we opt for some average over all the solutions of (15), to arrive at a single value for $\lambda$.

This feature-based technique provides parameters $\mathbf{R}_m$ and $\lambda \mathbf{c}_m$ that describe the motion of a current camera (and therefore point cloud) relative to the one preceding it. In order to place all the point clouds through the entire sequence in the same coordinate system, we initialize a rotation matrix $\mathbf{R}_{\text{tot}}$ to be the identity, and translation vector $\mathbf{c}_{\text{tot}}$ to be a zero vector. Then, once $\mathbf{R}_m$ and $\lambda \mathbf{c}_m$ have been estimated for a new point cloud in the sequence, we update them according to the following:

$$\mathbf{R}_{\text{tot}} \leftarrow \mathbf{R}_m \mathbf{R}_{\text{tot}}, \tag{16}$$
$$\mathbf{c}_{\text{tot}} \leftarrow \mathbf{R}_{\text{tot}}^T \lambda \mathbf{c}_m + \mathbf{c}_{\text{tot}}. \tag{17}$$

This total transformation is then applied to the new point cloud to bring it into the fixed coordinate system. A new point cloud arrives and the process is repeated: the motion of the new RGB camera relative to the previous one is estimated, scale is corrected, the total transformation is updated, and the new point cloud is transformed accordingly.

## V. Results

The methods described in this paper enable us to create a dense metric point cloud reconstruction, with colour assigned to each point, of a static scene. In this section we showcase some example reconstructions. It is difficult to quantitatively assess the accuracy of these results since independently generated "ground truth" is not available.

Results from two datasets are shown here. In capturing the first, a hand-held Kinect made a more-or-less full circle motion around a man in a chair. Some RGB images of this set are displayed in the top of Fig. 8. The figure also shows two partial reconstructions, each viewed from two angles, in the middle and bottom row. Unfortunately, some motion blurred RGB images about halfway through the sequence caused a glitch in the feature detection and matching, and the resulting erroneous motion estimation propagated through the rest of the sequence. Still, the set yielded decent and useful partial reconstructions as can be seen.

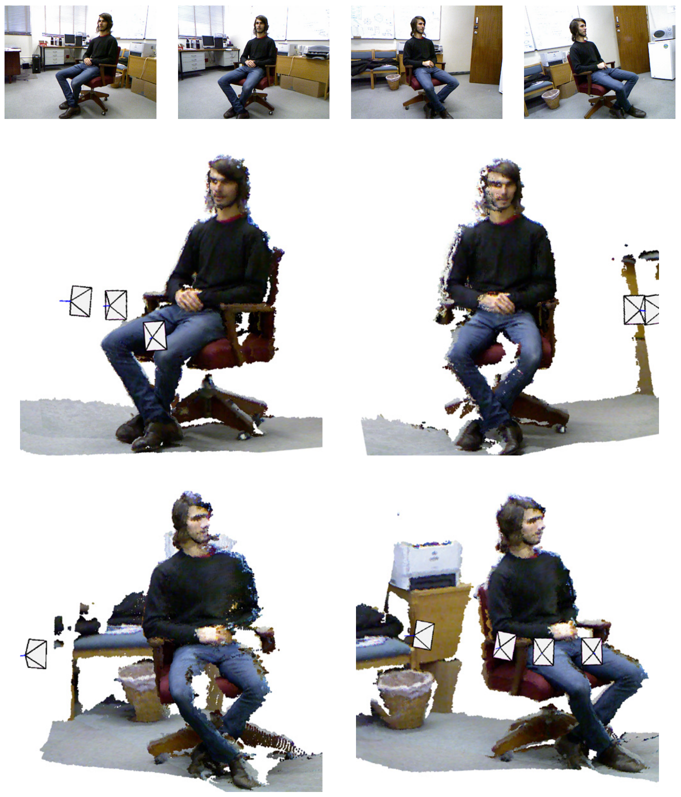

Fig. 8. Two partial reconstructions made from a captured dataset (the middle row shows two views of one, the bottom row two views of the other). Some of the RGB images of the dataset are shown on the top. Representations of some of the estimated camera positions can also be seen in the reconstructions.

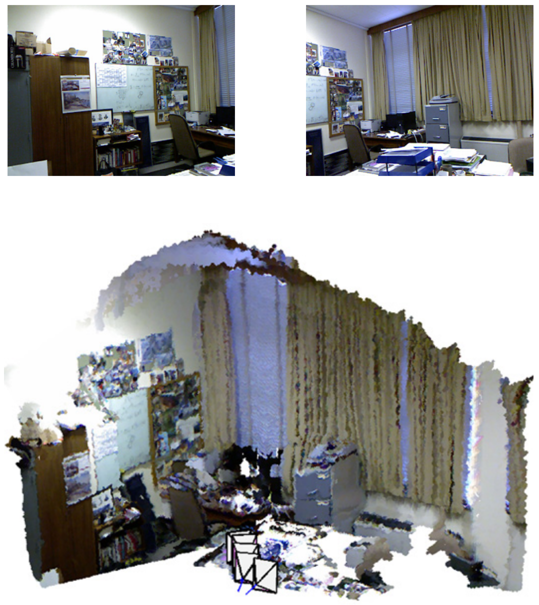

Fig. 9. A reconstruction made from a second dataset. Two of the RGB images are shown on top.

Fig. 9 shows images and a view of our reconstruction of the second dataset, taken of the interior of an office. Due to the cluttered nature of the scene this set yielded many feature matches and a good reconstruction was possible.

We note that the success of our alignment procedure hinges on the success of the feature matching step. Fig. 10 shows another part of the first dataset where inaccurate matching caused the alignment to fail. Again, as mentioned before, the moment a consecutive pair of RGB images delivers a small or largely incorrect set of matches, the estimated motion of the sensor becomes unreliable and destroys the rest of the reconstruction process. The fact that we take an incremental motion estimation approach causes a single error to propagate through the rest of the sequence.

Nevertheless, if good feature matching is achieved between consecutive RGB images, our methods can produce impressive 3D reconstructions.

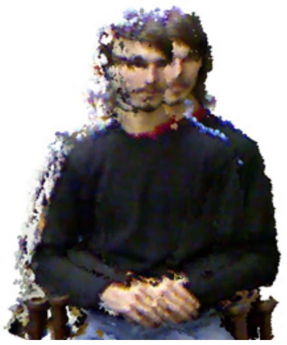

Fig. 10. This reconstruction shows the effect of inaccurate motion estimation, caused by too few correct feature matches, in the alignment process. The misalignment is clearly visible on the face and hands.

## VI. Conclusions and future work

In this paper we have successfully calibrated the IR and RGB cameras of a Kinect sensor, inferred a general mapping from its raw depth data to metric world coordinates, and mapped colours from a corresponding RGB image to the point cloud. This enabled us to create a metric point cloud reconstruction (measured in mm) with colour using a single depth and RGB image pair. A feature-based approach enabled us to align consecutive point clouds, formed from different uncontrolled viewpoints, in an effort to build a more complete dense reconstruction of a static scene. We argued that the success of this alignment procedure hinges on the quality of feature matches found between successive RGB images.

Some pre- and post-processing can be useful additions to the described techniques. Removing the "noise" in IR calibration images by some filter may aid towards more accurate calibration. We also note that the structured light technique used by the Kinect struggles around sharp edges (depth discontinuities) in a scene. Once the RGB camera has been successfully calibrated with the IR camera, colour information may become useful in dealing with noisy edges in the point cloud. We hope to further improve our motion estimation by the incorporation of bundle adjustment or the use of an estimator such as the Kalman filter. Our aligned reconstructions could also be used as an initialization for a more refined point cloud alignment procedure such as ICP [11], which in turn could feed back to our motion estimator.

## Acknowledgments

The authors thank Simon Muller for his technical assistance in capturing datasets, François Singels for clarifying some geometrical concepts, and Daniek Joubert for sharing her motion estimation code.

## References

[1] *The OpenKinect Project*, `http://openkinect.org`.
[2] V.I. Pavlovic, R. Sharma, T.S. Huang, *Visual interpretation of hand gestures for human-computer interaction: a review*, IEEE Pattern Analysis and Machine Intelligence, 7:677–695, 1997.
[3] J. Miura, Y. Negishi, Y. Shirai, *Mobile robot map generation by integrating omnidirectional stereo and laser range finder*, IEEE/RSJ International Conference on Intelligent Robots and Systems, 1:250–255, 2002.
[4] L.M. Lorigo, R.A. Brooks, W.E.L. Grimsou, *Visually-guided obstacle avoidance in unstructured environments*, IEEE/RSJ International Conference on Intelligent Robots and Systems, 1:373–379, 1997.
[5] R. Hartley, A. Zisserman, *Multiple View Geometry in Computer Vision, Second Edition*, Cambridge University Press, 2003.
[6] J. Bouguet, *Camera Calibration Toolbox for Matlab*, `http://www.vision.caltech.edu/bouguetj/calib_doc`.
[7] Z. Zhang, *A flexible new technique for camera calibration*, Technical Report MSRTR-98-71, Microsoft Research, 1998.
[8] R.C. Gonzalez, R.E. Woods, *Digital Image Processing, Third Edition*, Pearson Prentice Hall, 2008.
[9] D.G. Lowe, *Object recognition from local scale-invariant features*, Proceedings of the International Conference on Computer Vision, 2:1150-1157, 1999.
[10] M. Fischler, R. Bolles, *Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography*, Communications of the ACM, 24(6):381-395, 1981.
[11] P.J. Besl, N.D. McKay, *A method for registration of 3-D shapes*, IEEE Pattern Analysis and Machine Intelligence, 14(2):239–256, 1992.