RGB-D Tracking and Reconstruction for TV Broadcasts

Tommi Tykkälä[1], Hannu Hartikainen[2], Andrew I. Comport[3] and Joni-Kristian Kämäräinen[4]
[1]*Machine Vision and Pattern Recognition Laboratory, Lappeenranta University of Technology (LUT Kouvola), Lappeenranta, Finland*
[2]*Department of Media Technology, Aalto University, Aalto, Finland*
[3]*CNRS-I3S/University of Nice Sophia-Antipolis, Nice, France*
[4]*Department of Signal Processing, Tampere University of Technology, Tampere, Finland*

Keywords: Dense Tracking, Dense 3D Reconstruction, Real-time Tracking, RGB-D, Kinect.

Abstract: In this work, a real-time image-based camera tracking solution is developed for television broadcasting studio environments. An affordable vision-based system is proposed which can compete with expensive matchmoving systems. The system requires merely commodity hardware: a low cost RGB-D sensor and a standard laptop. The main contribution is avoiding time-evolving drift by tracking relative to a pre-recorded keyframe model. Camera tracking is defined as a registration problem between the current RGB-D measurement and the nearest keyframe. The keyframe poses contain only a small error and therefore the proposed method is virtually driftless. Camera tracking precision is compared to KinectFusion, which is a recent method for simultaneous camera tracking and 3D reconstruction. The proposed method is tested in a television broadcasting studio, where it demonstrates driftless and precise camera tracking in real-time.

1 INTRODUCTION

Rendering virtual elements, props and characters to live television broadcast combines augmented reality (AR) and video production. A robust camera estimation method is required for rendering the graphics from camera viewing direction. In film industry the process is known as *matchmoving* and it is traditionally done in post-processing (Dobbert, 2005). Matchmoving typically requires manual effort because the available tools are not fully automatic. In online broadcasting, special hardware based solutions exist for automatic camera tracking, but their value is degraded by limited operating volume, weaker reality experience and expensive price.

The goal of this work is to develop an affordable, portable and easy-to-use solution for television production studios. Marker-based AR techniques, such as ARToolKit (Kato and Billinghurst, 1999), are trivial to use but visible markers are irritating in the studio scene. Thus, we seek solution from more recent techniques which are able to track the camera without any markers. Visual simultaneous localisation and mapping (vSLAM) techniques are a true option, because they track the camera using a 3D model which is concurrently built based on visual measurements (Davison et al., 2007). Recently, low-cost RGB-D sensors have developed to a level where they can provide real-time stream of dense RGB-D measurements which are directly usable for camera tracking and scene reconstruction purposes.

In this work, we introduce an image registration based dense tracking and reconstruction method particularly for TV studios. Tracking precision is good, but over long sequences, time-evolving drift will eventually displace virtual props. We avoid drift by generating a RGB-D keyframe model, and tracking the camera relative to the nearest keyframe. The results are demonstrated with real studio shots. We thank Heikki Ortamo and Jori Pölkki for their professional support in a TV production studio.

1.1 Related Work

Traditionally vSLAM methods detect and track a sparse set of feature points which are matched in several images. The first feature-based visual SLAM methods used the extended Kalman filter (EKF) to update pose and structure (Davison et al., 2007). However, bundle adjustment has replaced EKF, because it is more accurate. PTAM (Klein and Murray, 2007) separated tracking and mapping into two par-

Tykkälä T., Hartikainen H., Comport A. and Kämäräinen J..
RGB-D Tracking and Reconstruction for TV Broadcasts.
DOI: 10.5220/0004279602470252
In *Proceedings of the International Conference on Computer Vision Theory and Applications* (VISAPP-2013), pages 247-252
ISBN: 978-989-8565-48-8

allel modules, where the mapping part was essentially bundle adjustment. To avoid feature extraction and matching problems, the raw pixel measurements can be used directly. DTAM was introduced as the dense version of PTAM, which allows every pixel to contribute to pose and structure estimation (Newcombe et al., 2011a). Finally, with KinectFusion system which replaces monocular camera by a RGB-D sensor (Newcombe et al., 2011b), pose tracking and structure estimation have become mature enough to be considered for live TV broadcasts.

In this work, we adopt the dense RGB-D approach (Tykkala et al., 2011; Audras et al., 2011; Comport et al., 2007). Our method differs rather strongly from the KinectFusion, because we estimate camera pose using dense RGB-D measurements instead of depth maps only. This is necessary because studio settings often contain planar surfaces for which the KinectFusion fails to track the camera. We do not estimate 3D structure concurrently because it can be solved prior to broadcasting. This simplification lightens computation and enables running the system with a low-end hardware. Robustness to outlier points is increased by selecting the photometrically stable pixels and by omitting unreliable regions (occlusion and moving objects) with the M-estimator.

2 DENSE TRACKING METHOD

2.1 Cost Function

Our dense pose estimation is defined as a direct color image registration task between the current image $I : \mathbb{R}^2 \Rightarrow \mathbb{R}$ and a frame $\mathcal{K}^* = \{\mathcal{P}^*, \mathbf{c}^*\}$, where $\mathcal{P}^* = \{\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_n\}$ is a set of 3D points and $\mathbf{c}^* = \{c_1, c_2, \ldots, c_n\}$ are the corresponding color intensities. Our goal is to find the correct pose increment $\mathbf{T}(\omega, \upsilon)$ which minimizes the following residual:

$$\mathbf{e} = I\left(w\left(\mathcal{P}^*; \mathbf{T}(\omega, \upsilon)\right)\right) - \mathbf{c}^*, \tag{1}$$

where $w(\mathcal{P}; \mathbf{T})$ is a projective warping function which transforms and projects $\mathcal{P}$ into a new view using the 4×4 transformation matrix $\mathbf{T}$ and the intrinsic matrix $\mathbf{K}$ (constant, omitted in the notation). The exact formula is

$$w(\mathcal{P}; \mathbf{T}, \mathbf{K}) = w\left(\mathcal{P}; \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}, \mathbf{K}\right) = N(\mathbf{K}(\mathbf{R}\mathcal{P} + \mathbf{t})), \tag{2}$$

where $N(\mathbf{p}) = (p_1/p_3, p_2/p_3)$ dehomogenizes a point. $\mathbf{T}(\omega, \upsilon)$ is defined as the exponential mapping which forms a Lie group $\mathbf{T}(\omega, \upsilon) \in \mathbb{SE}(3)$ (Ma et al., 2004):

$$\mathbf{T}(\omega, \upsilon) = e^{\mathbf{A}(\omega, \upsilon)}, \;\; \mathbf{A}(\omega, \upsilon) = \begin{bmatrix} [\omega]_\times & \upsilon \\ \mathbf{0} & 0 \end{bmatrix}, \tag{3}$$

where ω and υ are 3-vectors defining rotation and translation. From the practical point of view it is convenient to re-define (3) as

$$\mathbf{T}(\omega, \upsilon) = \widehat{\mathbf{T}} e^{\mathbf{A}(\omega, \upsilon)} \;\; , \tag{4}$$

which generates smooth increments around the base transform $\widehat{\mathbf{T}}$. This allows using iterative optimization and particularly *the inverse compositional trick* for estimating the transformation efficiently (Baker and Matthews, 2004).

2.2 Minimization

We adopt the inverse compositional approach for efficient minimization of the cost function (Baker and Matthews, 2004). The cost is reformulated as

$$\begin{aligned} \mathbf{c}^*(\omega, \upsilon) &= I^*\left(w\left(\mathcal{P}^*; e^{-\mathbf{A}(\omega, \upsilon)}\right)\right) \\ \mathbf{e} &= \mathbf{c}^*(\omega, \upsilon) - I\left(w\left(\mathcal{P}^*; \widehat{\mathbf{T}}\right)\right), \end{aligned}$$

where reference colors $\mathbf{c}^*$ are now a function of the inverse motion increment. The benefit is gained when computing the Jacobian

$$\mathbf{J}_{ij} = \Delta I^*\left(w(\mathcal{P}_i; \mathbf{I})\right) \frac{\partial w(\mathcal{P}_i; \mathbf{I})}{\partial x_j}, \tag{5}$$

where $\Delta I(\mathbf{p}) = [\, \frac{\partial I(\mathbf{p})}{\partial x} \;\; \frac{\partial I(\mathbf{p})}{\partial y} \,]$. By this trick $\mathbf{J}$ does not depend on the current transformation anymore and it is sufficient to computed it only once for each $\mathcal{K}^*$. Gauss-Newton iteration is well-suited for minimization when the initial guess is near the minimum and the cost function is smooth. The cost function is locally approximated by the first-order Taylor expansion $\mathbf{e}(\mathbf{x}) = \mathbf{e}(\mathbf{0}) + \mathbf{J}\mathbf{x}$, where $\mathbf{e}(\mathbf{0})$ is the current residual. Now the scalar error function becomes

$$\frac{1}{2}\mathbf{e}(\mathbf{x})^T \mathbf{e}(\mathbf{x}) = \frac{1}{2}\mathbf{e}(\mathbf{0})^T \mathbf{e}(\mathbf{0}) + \mathbf{x}^T \mathbf{J}^T \mathbf{e}(\mathbf{0}) + \frac{1}{2}\mathbf{x}^T \mathbf{J}^T \mathbf{J}\mathbf{x} \tag{6}$$

where the derivative is zero when

$$\mathbf{J}^T \mathbf{J}\mathbf{x} = -\mathbf{J}^T \mathbf{e}(\mathbf{0}) \;\; . \tag{7}$$

Because $\mathbf{J}^T\mathbf{J}$ is a 6×6 positive definite matrix, the inversion can be done efficiently using Cholesky method. The matrix multiplication associativity in $\mathbb{SE}(3)$ enables collecting the previous increments into the base transform by $\widehat{\mathbf{T}_k} \cdots \widehat{\mathbf{T}_1} e^{\mathbf{A}(\omega, \upsilon)} \Rightarrow \widehat{\mathbf{T}} e^{\mathbf{A}(\omega, \upsilon)}$ (4).

2.3 Keyframe based Reference $\mathcal{K}^*$

Minimisation of the cost function (1) with the Gauss-Newton method in Sec 2.2 provides us real-time dense RGB-D camera tracking (and scene reconstruction), which frame by frame finds the optimal transformation $\mathbf{T}(\omega,\upsilon)$ between a previous frame $\mathcal{K}^*$ and a current frame (image I). This approach has one disadvantage: small per frame estimation error cumulates into global drift which gradually displaces virtual elements. Fortunately, the studio scene can be pre-recorded into a set of keyframes prior to broadcasting (see Fig. 1). The tracking is then defined relative to the nearest keyframe $\mathcal{K}^* \leftarrow \text{select\_closest}(\mathcal{K}_j^*)$. For small studios, a single dense tracking sweep is sufficient, but methods exist also for building larger models (Henry et al., 2012).

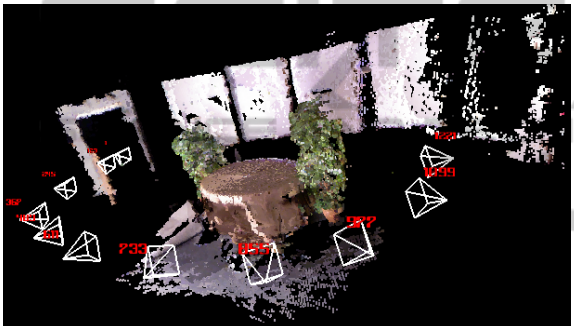

Figure 1: Stored scene keyframes and their pixels mapped to a common 3D world model.

2.4 Robust Estimation

Now we know that our reference keyframes contain merely static geometry with Lambertian reflection, it is possible to increase robustness of the estimation by emphasising stronger color and depth correlation between the static reference points and the current 3D points matched in each iteration. We do this by employing M-estimation in which uncertainty based weights $\mathrm{w_k} \in [0,1] \in \mathbb{R}$ are given to the residual elements e_k. Color correlation is enforced by the Tukey weighting function

$$u_k = \frac{|e_k|}{c * median(\mathbf{e}_a)}, \tag{8}$$

$$\mathrm{w_k^c} = \begin{cases} (1-(\frac{u_k}{b})^2)^2 & \text{if } |u_k| <= b \\ 0 & \text{if } |u_k| > b \end{cases}, \tag{9}$$

where $\mathbf{e}_a = \{\|e_1\|,\ldots,\|e_n\|\}$, $c = 1.4826$ for robust standard deviation, and $b = 4.6851$ is the Tukey specific constant.

The depth correlation weights are computed simultaneously by a depth lookup

$$\mathbf{e}_z = \mathcal{Z}(w(\mathcal{P};\mathbf{T}(\omega,\upsilon))) - \mathbf{e}_3^T\mathbf{T}(\omega,\upsilon)\begin{bmatrix}\mathcal{P}\\1\end{bmatrix}, \tag{10}$$

where $\mathcal{Z}:\mathbb{R}^2 \Rightarrow \mathbb{R}$ is the depth map function of the current RGB view and $\mathbf{e}_3^T = (0,0,1,0)$ selects the depth coordinate. When the standard deviation of depth measurements is τ, the warped points whose depth differs more than τ from the current depth map value can be interpreted as foreground actors/outliers. Thus we define a diagonal matrix

$$\mathrm{w_k^z} = max\left((1-\mathbf{e}_z^2(k)/\tau^2),0\right)^2, \tag{11}$$

The weighted Gauss-Newton step is obtained by re-writing (7)

$$\mathbf{J}^T\mathbf{W}\mathbf{J}\mathbf{x} = -\mathbf{J}^T\mathbf{W}\mathbf{e}, \tag{12}$$

where $\mathbf{W}$ is a diagonal matrix with $\mathrm{diag}(\mathbf{W})_k = \mathrm{w_k}^c\mathrm{w_k}^z$. The robust step is obtained by $\mathbf{J} \Leftarrow \sqrt{\mathbf{W}}\mathbf{J}$ and $\mathbf{e} \Leftarrow \sqrt{\mathbf{W}}\mathbf{e}$. The both weight components are quadratic and therefore the square roots are not evaluated in practise.

2.5 Selecting Keyframe Points

The reference points $\mathcal{P}^*$ can be freely selected from the keyframe. When considering the minimization of the photometrical error, only the 3D points with image gradients infer pose parameters. Therefore the majority of the points can be neglected based on the magnitude of the image gradient. Gradient vector $\nabla\mathbf{c}$ is evaluated at the projection of $\mathcal{P}^*$ using bilinear interpolation. Instead of sorting $|\nabla\mathbf{c}|$, we generate the histogram of the magnitudes. The value range is bounded to $[0,255]$. We seek the bin B_t for which $\Sigma_{k=B_t-1}^{255} h(k) < n <= \Sigma_{k=B_t}^{255} h(k)$, where n is the number of points to be selected.

3 IMPLEMENTATION DETAILS

Our method was implemented in Ubuntu Linux environment using open software tools, Kinect RGB-D sensor and a commodity PC laptop hardware on which the method runs real-time. The main limitations are the operating range ($\approx 1m-5m$) and that it uses controlled IR lighting which may not work in outdoors. Below, we discuss Kinect related implementation issues.

3.1 Kinect Calibration

Since Microsoft Kinect can be set into a special mode where the raw IR images can be stored, it is possible to use standard stereo calibration procedure for obtaining the calibration parameters for the IR and the RGB view (Bouguet, 2010) (Fig. 2). IR view and the depth view are trivially associated with image offset $(-4,-3)$.

A single camera calibration is used to initialize IR and RGB camera parameters. Then calibration is followed by a stereo procedure which re-estimates all free parameters ($\mathbf{K}_{IR}$, $\mathbf{K}_{RGB}$, $\mathbf{kc}_{RGB}$,$\mathbf{T}_b$). The lens distortion parameters of the IR camera are forced to zero, because data has already been used to generate the raw disparity map. This means that the IR lens distortion is compensated by tweaking other parameters. The RGB camera lens distortion parameters $\mathbf{kc}_{RGB}$ are estimated without any special concerns. In practice, the distortion seems to be minor and the first two radial coefficients are sufficient ($\mathbf{kc}_{RGB} = (0.2370,-0.4508,0,0,0)$, and the stereo baseline is $b = 25.005mm$. $\mathbf{T}_b$ stores the baseline transform as 4×4 matrix. The conversion from raw disparities into depth values can be done by $z = \frac{8pf}{B-d}$, where p is the baseline between the projector and the IR camera, B is a device specific constant and f is IR camera focal length in pixel units. p and B are estimated by solving the linear equation $\begin{bmatrix}-\mathbf{1} & \mathbf{Z}\end{bmatrix}\begin{bmatrix}A & B\end{bmatrix}^T = \mathbf{D}$, where $A = 8pf$, $\mathbf{Z}$ is $n\times1$ matrix of reference depth values z_k from the chessboard pattern (Caltech calibration), and $\mathbf{D}$ is a $n\times1$ matrix whose elements are $d_k z_k$. The parameters will be $p \approx 75mm$ and $B \approx 1090$. Note, that this reconstruction method is merely an approximation which procludes measurements at long ranges. There are also dedicated calibration toolboxes for RGB-D sensors, which model disparity distortion accurately (Herrera et al., 2012).

Figure 2: RGB and IR images of the calibration pattern.

3.2 Dense Tracking with Kinect

The reference point clouds $\{\mathcal{P}^*,\mathbf{c}^*\}$ in (1) were generated from Kinect RGB image and disparity map in the following way. Kinect Bayer images were converted into RGB format, downsampled into 320x240 and undistorted from lens distortions. Downsampling is almost lossless due to sparsity of RGB values in the Bayer pattern. The raw disparity map was first converted into a depth map, downsampled into 320x240 size using max filter and then transformed into a point cloud $\mathcal{P}^*_{IR}$. Maximum filtering is chosen because it does not produce artificial geometry. $\mathcal{P}^*$ was then generated from $\mathbf{T}_b\mathcal{P}^*_{IR}$, where $\mathbf{T}_b$ is the baseline transformation between the IR and RGB cameras. Points $\mathbf{p} \in \mathcal{P}^*$ do not exactly project to the pixel centers of the RGB image grid, and thus, bi-linear interpolation is used for generating the corresponding intensities $\mathbf{c}^*$.

The cost function is minimized using coarse-to-fine approach using image pyramid with 80×60, 160×120 and 320×240 layers for each RGB-D input frame.

4 EXAMPLES

4.1 Dense Tracking vs Kinfu

Kinfu is the open source implementation of KinectFusion (Rusu and Cousins, 2011). We compare our dense tracking accuracy with Kinfu using the RGB-D SLAM benchmark provided by Technical University of Münich (Sturm et al., 2011) (Figure 3). Dense tracking is executed incrementally by using a recent view as the reference. Thus, both methods aim at tracking the camera pose without a prior model and small drift will be present. The major difference between the systems is that Kinfu optimizes a voxel based 3D structure online and uses the iterative closest point (ICP) approach for pose estimation. Kinfu has small drift when the voxel size is small and geometry is versatile. Kinfu fails in bigger operating volumes, because voxel discretization becomes coarse and, for these sequences especially, the volume will also contain planar floor which break downs ICP[1]. In broadcasting studios, scenes are often larger than $(3m)^3$ and geometrical variations can not be guaranteed. Our dense tracking demonstrates robust tracking even when planar surfaces are present, because the cost function matches also scene texturing. Memory consumption is low even in larger operating volumes, because RGB-D keyframes can be memory-optimized based on the viewing zone.

Table 1 shows the comparison between our method and Kinfu numerically. The dense tracking drifts $1.08cm/s$ with the slower `freiburg2/desk` sequence and $2.60cm/s$ with the faster `freiburg1/desk` sequence. Our dense tracking has smaller error when the camera is moving faster. Kinfu

[1]Video: http://youtu.be/tNz1p1sdTrE

Table 1: Drift is evaluated using standard sequences with known ground truth trajectories. The proposed approach has smaller drift when a camera is moving faster. The keyframe model can be memory-efficiently built. Kinfu has smaller drift when a camera is moving slower but bigger scenes are not possible due to inscalability of the voxel grid. Problems exist with planar surfaces. The computational requirements are higher even though Kinfu is executed on a powerful desktop hardware.

Dataset	Our drift	Kinfu(3)	Kinfu(8)	Camera speed
`freiburg1/desk`	$2.60cm/s$	$8.40cm/s$	$3.97cm/s$	$41.3cm/s$
	$52.2ms$	$135ms$	$135ms$	
`freiburg2/desk`	$1.08cm/s$	$0.64cm/s$	$1.30cm/s$	$19.3cm/s$
	$35.5ms$	$135ms$	$135ms$	

has lower drift when using $(3m)^3$ voxel grid, but fails to operate in bigger volumes $(8m)^3$ that match with broadcasting studios. Also the computational requirements of Kinfu are significantly higher compared to our approach even though Kinfu is executed on a powerful desktop hardware. Drift was measured by dividing the input frames into subsegments of several seconds (10 and 2 correspondingly) whose median error was measured against the ground truth. 1 second average error was computed from the median subsegment. The error values are computed from bigger windows to average out random perturbations and neglect gross tracking failures which occur with Kinfu in all cases except on `freiburg2/desk` using $(3m)^3$ volume.

4.2 Driftless Keyframe Tracking

The relative poses between the keyframes could, in theory, be obtained by bundle adjustment techniques (Triggs et al., 2000) if feature point extraction and matching succeeded, and a good initial guess would exist. With the studio sequences recorded, texturing was so limited that popular tools, such as `Bundler` (Snavely et al., 2006) failed without manual annotated feature points. As an alternative solution, we utilised the proposed dense RGB-D tracker to incrementally build a keyframe model. It is noteworthy, that a single quick sweep of the scene produces accurate keyframe model[2]. Figure 4 illustrates how dense tracking is used to sweep a keyframe model of the studio scene rapidly. With pre-recorded keyframes, the online broadcasts are guaranteed to operate with the correct 3D geometry. $\mathcal{K}_j^*$ were selected by picking RGB-D frames evenly from the recorded sequence. A keyframe database is illustrated in Figure 1.

In Figure 5, we show how the drift increases in a studio environment when using dense tracking. On the right-hand side, the drift problem is solved by tracking relative to keyframes[3]. Note that in long-term use, even a small number of keyframes eventually outperforms the dense tracking due to drift. However, a small keyframe number produces drift jumps between the keyframes which can be visually disturbing. With sufficient number of keyframes, the error remains small.

[2]Video: http://youtu.be/wALQB3eDbUg

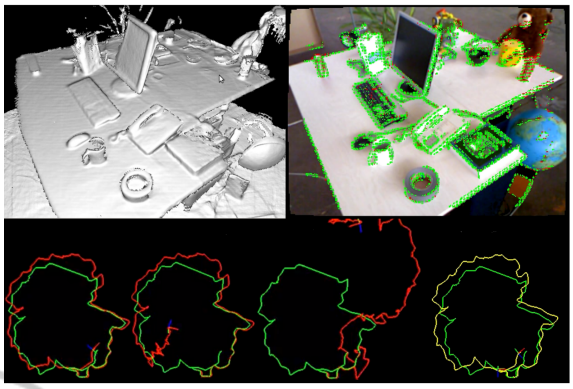

Figure 3: On the left Kinfu result for `freiburg_desk2` sequence using 3x3x3, 5x5x5 and 8x8x8 meter voxel volume. Kinfu gains lower drift due to structural optimization, but planar surfaces cause tracking failures. 3x3x3 volume does not contain floor and therefore Kinfu works well. Limited operating volume is a problem for practical use in the studio. On the right, our dense tracking drift illustrated when a fixed keyframes are not used. Problems with planar surfaces or limited operating volume do not exist. Green dots reveal the selected points for given RGB-D measurement.

Figure 4: Camera trajectory solved from Kinect input by minimizing the cost function (1). A small shark is rendered into the studio scene.

5 CONCLUSIONS

In this work, camera pose tracking was defined as a photometrical registration problem between a refer-

[3]Video: http://youtu.be/zfKdZSkG4LU

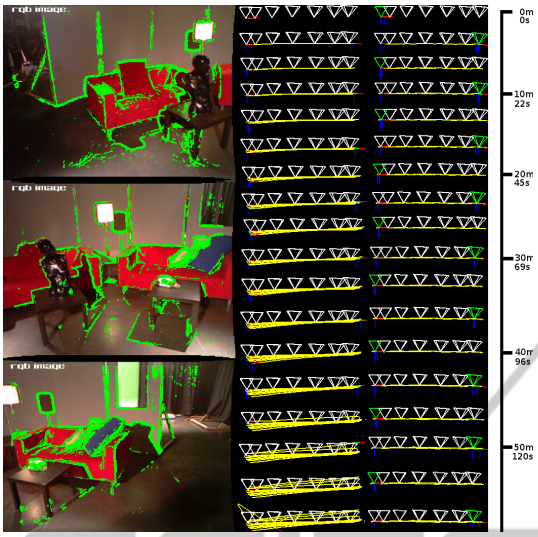

Figure 5: Camera moving front and back along a fixed 3.30m studio rail. On the left, three images taken from the beginning, middle and the end of the rail. Green regions illustrated selected points. On the right, comparison between dense tracking and keyframe tracking. In dense tracking drift increases in time, but keyframe tracking maintains small bounded error. A person is moving in the scene during the last four cycles.

ence frame and the current frame. To remove the global drift in incremental tracking, the closest prerecorded keyframe was chosen to be the motion reference. The system was designed to be an affordable solution for TV broadcasting studios relying only on the Kinect sensor and a commodity laptop. The proposed approach performs robustly in a standard benchmark, where KinectFusion has problems with planar surfaces and limited voxel grid resolution. Our future work will address the practical issues how studio staff and camera men can use our computer vision system in live broadcasts. Moreover, combination of the best properties of our approach and KinectFusion will be investigated.

REFERENCES

Audras, C., Comport, A. I., Meilland, M., and Rives, P. (2011). Real-time dense rgb-d localisation and mapping. *In Australian Conference on Robotics and Automation. Monash University, Australia, 2011*.

Baker, S. and Matthews, I. (2004). Lucas-kanade 20 years on: A unifying framework. *Int. J. Comput. Vision*, 56(3):221–255.

Bouguet, J.-Y. (2010). Camera calibration toolbox for matlab. http://www.vision.caltech.edu/bouguetj/calib_doc.

Comport, A., Malis, E., and Rives, P. (2007). Accurate quadri-focal tracking for robust 3d visual odometry. In *IEEE Int. Conf. on Robotics and Automation, ICRA'07*, Rome, Italy.

Davison, A., Reid, I., Molton, N., and Stasse, O. (2007). MonoSLAM: Real-time single camera SLAM. *PAMI*, 29:1052–1067.

Dobbert, T. (2005). *Matchmoving: The Invisible Art of Camera Tracking*. Sybex.

Henry, P., Krainin, M., Herbst, E., Ren, X., and Fox, D. (2012). RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments. *The International Journal of Robotics Research*, 31(5):647–663.

Herrera, C., Kannala, J., and Heikkila, J. (2012). Joint depth and color camera calibration with distortion correction. *IEEE PAMI*, 34(10).

Kato, H. and Billinghurst, M. (1999). Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *Proceedings of the 2nd International Workshop on Augmented Reality (IWAR 99)*, San Francisco, USA.

Klein, G. and Murray, D. (2007). Parallel tracking and mapping for small ar workspaces. *Proceedings of the International Symposium on In Mixed and Augmented Reality (ISMAR)*, pages 225–234.

Ma, Y., Soatto, S., Kosecka, J., and Sastry, S. (2004). *An invitation to 3-D vision: from images to geometric models*, volume 26 of *Interdisciplinary applied mathematics*. Springer, New York.

Newcombe, R., Lovegrove, S., and Davison, A. (2011a). Dtam: Dense tracking and mapping in real-time. In *ICCV*, volume 1.

Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohli, P., Shotton, J., Hodges, S., and Fitzgibbon, A. (2011b). Kinectfusion: Real-time dense surface mapping and tracking. *ISMAR*, pages 127–136.

Rusu, R. B. and Cousins, S. (2011). 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China.

Snavely, N., Seitz, S. M., and Szeliski, R. (2006). Photo tourism: Exploring photo collections in 3d. In *ACM TRANSACTIONS ON GRAPHICS*, pages 835–846. Press.

Sturm, J., Magnenat, S., Engelhard, N., Pomerleau, F., Colas, F., Burgard, W., Cremers, D., and Siegwart, R. (2011). Towards a benchmark for rgb-d slam evaluation. In *Proc. of the RGB-D Workshop on Advanced Reasoning with Depth Cameras at Robotics: Science and Systems Conf. (RSS)*, Los Angeles, USA.

Triggs, B., McLauchlan, P., Hartley, R., and Fitzgibbon, A. (2000). Bundle adjustment - a modern synthesis. In Triggs, B., Zisserman, A., and Szeliski, R., editors, *Vision Algorithms: Theory and Practice*, volume 1883 of *Lecture Notes in Computer Science*, pages 298–372. Springer-Verlag.

Tykkala, T. M., Audras, C., and Comport, A. (2011). Direct iterative closest point for real-time visual odometry. In *ICCV Workshop CVVT*.