RKLT: 8 DOF real-time robust video tracking combing coarse RANSAC features and accurate fast template registration

Xi Zhang, Abhineet Singh, Martin Jagersand
Dept of Computing Science
University of Alberta
Edmonton, Canada
Email: {xzhang6,asingh1}@ualberta.ca, jag@cs.ualberta.ca

***Abstract*—The performance of a tracker can be measured by two often conflicting criteria - robustness and accuracy. Recently researchers have focused on improving robustness, using adaptive appearance models. However updating the appearance model can cause drift and lower the accuracy of motion (state) estimation. These trackers generally compute 2 degree of freedom(DOF) image translation of the object, and are suited for applications such as surveillance. In contrast, we are interested in tracking objects using high DOF motion models - especially 8DOF homography models that allow tracking of precise state information (projective 8D or calibrated 3D world translations and 3D rotations of the tracked object). Such precise state is required for visual motion control of e.g. robot arms, hands and UAV.**

To this end, we propose a novel tracking algorithm that combines KLT [8], RANSAC [21] and Inverse Compositional tracker [7]. First we sample a large patch into a set of small patches and track each one using frame-to-frame 2D KLT trackers. An 8 DOF homography describing the large patch motion is then estimated from the current locations of these KLT trackers using RANSAC while also discarding lost trackers as outliers. Finally, using the RANSAC 8DOF motion estimate as the initial guess, we perform a few iterations of an IC registration tracker. This refines the patch motion to sub pixel accuracy and avoids drift by registering to the original template. We perform three sets of experiments - one is the standard synthetic Lena convergence benchmark and two use real image sequences from recent datasets - to show that our tracker compares favorably with the state-of-the-art.

***Keywords*-Registration based tracker; RANSAC; KLT; Evaluation methods.**

I. Introduction

Visual tracking is one of the most important and well researched topics in computer vision. Breakthroughs in machine learning have given rise to numerous new tracking algorithms in recent times - over twenty trackers published in the last decade were compared in [11]. Most of these trackers, however, fall in the class of 'learning based trackers' that rely on building adaptive appearance models that are robust to changes in the object's appearance due to events like partial occlusions and lighting variations. Though these are suitable for long term tracking applications like surveillance, where not losing track of the object is more important than maintaining a highly accurate estimate of its location, they do not perform as well in many robotics and virtual reality applications which involve deft object manipulations that require very precise information about its location and orientation. For example, the motion states estimated by [13], [14] have only 3 DOF for handling translation and scaling and even the 6 DOF affine model used in [12] [15] is rarely good enough to provide sub-pixel accuracy for complex motions.

This is the reason that another class of trackers - called registration based trackers - have been more successful in the domain of high precision tracking. Unlike learning based trackers, these work on the assumption that the object is always fully or partially visible and any changes in its appearance are due to the warping (i.e. geometric transformation) induced by relative motion between the object and the camera. The goal of these trackers is thus to estimate the optimal warping parameters in each frame such that when the original template is warped using these parameters, the resulting patch in the current frame is visually similar to the template. Trackers within this class differ mainly in the search methods and motion models they employ. Most trackers use some approximation to gradient descent search methods like Newton, Gauss-Newton and Levenberg-Marquardt search [1]. In contrast, our algorithm searches using RANSAC [21] - a parameter estimation and outlier detection method.

Motion models codify assumptions about how the object moves from frame to frame as well as how complex its global motion is allowed to be. The latter is quantified by the degrees of freedom (DOF) of the geometric transform that gives its warped position. For instance, an object that can only translate in the image plane has 2 DOF motion while one that can also undergo in-plane rotations, shearing and scale changes (due to out of plane translations) has 6 DOF. The latter, called affine model, is sufficient for most practical purposes. However, taking into account out-of-plane rotations can be crucial for the success of dexterous tasks involving fine alignments and manipulations of planar objects and surfaces in 3D. This is where the 8 DOF homography model is important since this is the only one that accurately models the image projection of planar

978-1-4799-1986-4/15 $31.00 © 2015 IEEE
DOI 10.1109/CRV.2015.18

objects (using a perspective camera model). Though several registration based trackers use this model [1] [2] [3], they are often very susceptible to drift and failure due to the inherent complexity of estimating so many free parameters. We show through experiments that our 8 DOF estimator is more accurate and less prone to failure than the best of these trackers.

In this paper (Section III) we present a tracker which aims to combine the robustness of RANSAC on inter-frame feature motion with the accuracy of global template-based registration. Our algorithm tracks a planar homography between a selected template and later image frames in two steps. First the template is subdivided, and multiple KLT trackers used to track motion between consecutive frames. RANSAC on the KLT features provides an (approximate) homography. Second an Inverse Compositional (IC) global registration is performed between between the template from the first frame and the current frame. In this second registration step, convergence is significantly improved over standard IC by using the RANSAC estimate to start numerical Gauss-Newton (G-N) search close to the solution, and by only solving the G-N equations for the RANSAC inlier pixels.

In experiments (Section IV), we test convergence of our method on the standard Lena benchmark [1], [2], [3], the Metaio poster tracking benchmark [4], and the TMT dataset [6]. The experiments show that our RKLT tracker tracks significantly better than the base IC algorithm alone [7], or KLT+RANSAC alone. It is also better than ESM [2], and NNIC [3], the two leading methods in 8DOF tracking according to previously published benchmarks [4], [6].

II. Related work

Conventional registration based trackers [1] [2] use some approximation to gradient descent search to find the optimal warp parameters. Several trackers of this type have been compared in [1]. In contrast to earlier works that used Newton type search methods, [2] proposed an efficient second order method that eliminated the costly Hessian computation needed in Newton type methods while also achieving high convergence rate. This tracker is often considered as the state-of-art in registration based tracking.

Instead of using gradient information, Dick, et al [3] proposed an alternative approach that takes advantage of the efficient Approximate Nearest Neighbur(ANN) search algorithm to search among a large set of precomputed warped patches to find one that closely matches the current warped patch. Using the popular FLANN library [10], this can be accomplished very quickly which allows an extra refinement step without sacrificing real time performance. This is quite similar to the approach used in this work - both methods generate good initial guesses that can be refined using Lucas Kanade type search to achieve sub pixel accuracy. However, unlike RANSAC used in our work, ANN search has the significant limitation of needing an initial training stage which limits its applicability to scenarios that require fast initialization.

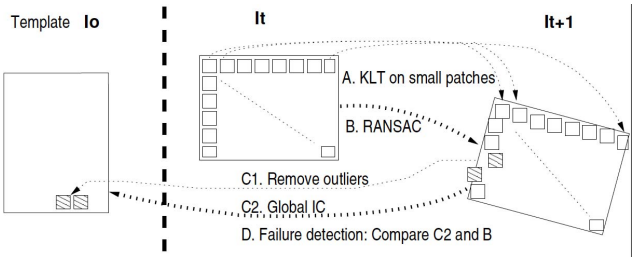

Figure 1. RKLT tracker overview. Tracking image template is selected in I_0. (A) Several KLT trackers compute 2D image motion. (B) RANSAC estimates a frame-to-frame homography. (C1) RANSAC outlier regions in the template are removed (shaded patches). (C2) Inverse Compositional (IC) global homography registration in the original template coordinate frame. (D) If the IC incremental update computation diverges (significantly different from RANSAC H), then global warp H updated by RANSAC, otherwise the more precise IC H is returned.

Another tracker closely related to our work is the median flow tracker [18], which was later adapted for the Tracking-Learning-Detection framework [13]. Similar to our work, a set of evenly sampled points are initialized within the object bounding box in the first frame and then tracked using Lucas Kanade trackers. However, lost trackers are rejected using the Forward-Backward error criteria instead and the remaining points are used to estimate only translation and scaling of the patch. This limits its applicability to robotics and VR applications since it can only handle 3 DOF motion. Though our algorithm is indeed inspired by this method, we have improved upon it by extending it to handle 8 DOF motion while also using a more robust rejection criterion.

The idea of using RANSAC to estimate the homography between two frames from sets of matching points was also used by [19] but in the context of panoramic image stitching. It was the good results reported there that encouraged us to apply this method for high DOF tracking. However, one limitation in our case is that [19] has used feature detection to find good points to match which is not suitable in our tracking framework since the objects we typically have to track are often small and/or have low textured appearance making it difficult to extract enough feature points to get a good estimate.

III. Methodology

The system overview can be found in Fig. 1. Our complete algorithm, given in Algo. 1, can be divided into the following four main steps:

A. Sampled Points Tracking

As mentioned earlier, the detection and matching based frame work in [19] can not be directly applied to tracking. Instead, we have used a similar idea as Kalal et. al did in their median flow tracker [18] where evenly sampled point

trackers were initialized in each frame and tracked in the next. Instead of applying any further filtering step as in [18], all the point pairs between the adjacent frames are fed into the robust RANSAC algorithm. Though any reasonably fast tracker can be used for this step, we have used the pyramid KLT point trackers [8] in our work for convenience since an efficient and bug free implementation is available in OpenCV [22].

B. Homography Estimation

Since we are using simple coarse trackers in the above step, it is likely that some of the trackers we initialized in this step will not track well due to factors like lighting variations, occlusions or simply large inter frame motions. Trackers like [7] and [2] that track the object as a single template may also face this problem. In fact, since they formulate the search as an L2 minimization problem, they may be even more sensitive to these factors. However, in our case, since each point tracker only tracks a very small patch and does this only for one frame (as the trackers are reinitialized in each frame), it is a fair assumption that most points will be successfully tracked. We consider the initial and final positions of these trackers as pairs of matching points that are used as input to the robust RANSAC algorithm [21] to estimate the homography that best describes their relative positions. Since RANSAC detects outliers as part of the estimation process, the lost trackers are implicitly discarded in this step.

Let I_t denote the frame at time t and H_t denote the homography that warps the initial template in I_0 into the object patch in I_t. Then, if H_t^{t+1} is the true homography between I_t and I_{t+1}, we can get H_{t+1} by Eq. 1.

$$H_{t+1} = H_{t+1}^{t} H_t \tag{1}$$

Therefore, if $\hat{H}_t^{t+1}$ is the estimate of the homography between I_t and I_{t+1} produced by RANSAC, we can get a coarse estimate of H_{t+1} by Eq. 2.

$$\hat{H}_{t+1} = \hat{H}_{t+1}^{t} H_t \tag{2}$$

In general, $\hat{H}_t^{t+1}$ will be close but not equal to H_t^{t+1} unless all the points are tracked perfectly in Step A. Hence, without the following refinement step, the accumulated error when estimating H_{t+1} will keep increasing and will eventually cause tracking drift. This why an extra step is needed to refine $\hat{H}_t^{t+1}$ and get it closer to H_t^{t+1}.

C. Tracking Refinement

The goal of this step is to find a better estimate for H_t^{t+1} given the coarse estimate $\hat{H}_t^{t+1}$ from Step B. This is achieved by using $\hat{H}_{t+1}$ as the initial guess or starting point for the search process of an efficient gradient based tracking algorithm that was initialized with the original template in I_0. We have used the IC tracker [7] for this purpose since this is one of the faster variants of the Lucas Kanade tracker. Since we are assuming that our initial guess $\hat{H}_t^{t+1}$ is a good one, far fewer iterations should be required for convergence than if we were to use H_t as the starting point for the search.

A further speedup in this step is achieved by considering only a part of the full template, determined by the inliers from Step B, while solving the optimization equations of the IC tracker. In other words, we sample the pixels in the template and only pass those that were successfully tracked in Step A to the IC tracker. This is based on the assumption that the points where the trackers in Step A failed are probably the ones that are most difficult to track and hence are least likely to contribute to the convergence of the IC tracker since both trackers use different variants of approximate gradient descent search. This is an additional reason for using gradient descent based trackers for both steps A and C, even though, in principle, any two trackers can be used.

An important point to note here is that the intensity image template for the IC tracker, unike the KLT point trackers in Step A, is *not* reinitialized at each frame, but the original template is used throughout the tracking task. Therefore, this step registers the current image with the first template, and removes the drift common in trackers that update/learn online the target appearance model during the tracking.

D. Failure Detection

Since the tracker chosen in Step C has to be fast, the choice of this tracker is determined more by considerations of efficiency and accuracy than robustness. Therefore, even if the initial guess from Step B is good, this tracker may still be affected by noise and fail. For this reason, we use a simple way to detect failure in Step C by measuring the alignment error err (defined in Equation 3) of the four corners of the bounding boxes resulting from steps B and C. We empirically selected a threshold th such that, when $err >= th$ we assume that Step C failed and the final tracking results will be the one given by Step B (i.e. $\hat{H}_{t+1}$).

Algorithm 1

$H_1 \leftarrow$ *Identity Matrix*
$X_1 \leftarrow$ *Evenly sampled points from template*
for *each new frame* I_i **do**
  for each x_j in X_i **do**
    Initialize a point tracker K_j with x_j on I_i
    $\hat{X}_{i+1} \leftarrow K_j$ *result on* I_{i+1}
  $\hat{H}_i^{i+1} \leftarrow$ *RANSAC* $(X_i, \hat{X}_{i+1})$
  $\hat{H}_{i+1} \leftarrow \hat{H}_i^{i+1} H_i$
  $H_{i+1} \leftarrow$ *Apply refinement tracking algorithm given* $\hat{H}_{i+1}$
  $X_{i+1}, H_{i+1} \leftarrow$ *FailureDetection* $(\hat{H}_{i+1}, H_{i+1})$

IV. Experiments

In this section, we evaluate our trackers with two different types of experiments: the static image experiments using synthetically generated Lena images as well as experiments on real datasets. In [11], Wu, et al defined two error metrics for tracker evaluation: the center location error and the bounding box overlap between tracked results and the ground truth. However, both of these are aimed to measure the robustness of trackers and thus give relatively low importance to accuracy. Meanwhile, in a dataset released by Lieberknecht, et al [4] and a later one by Roy, et al [6], a more suitable error metric for registration based trackers was introduced as in Eq. 3.

$$MCD(c, gt) = \sqrt{\frac{1}{4}\sum_{j=1}^{4} ||c_j - gt_j||_2^2} \tag{3}$$

Where $c_j \subset \mathbb{R}^2$ represent the four corners of the tracked object bounding box in the current frame and $gt_j \subset \mathbb{R}^2$ are the corresponding locations in the ground truth. We have used this error metric as the basic performance measure in our experiments and refer to it as MCD(Mean Corner Distance). In addition, we have adopted two evaluation methods which were introduced in [6]: average drift and success rate. The former is mainly designed for measuring accuracy and the latter for robustness. For our experiments, we have reformulated these as follows:

Average Drift: Given a sequence F with n frames, the average drift of a tracker is defined as the average MCD of those frames that have been successfully tracked, i.e. whose MCD is below the threshold *TH* pixels (Eq. 4).

$$AD(C, GT, K) = \frac{\sum_{k \in K} MCD(C_k, GT_k)}{\mathbf{card}(K)}, \tag{4}$$

where the index set K is defined as $K = \{k : MCD(C_k, GT_k) \leq TH\}$ while C and GT respectively contain c and gt for all n frames. $\mathbf{card}(\mathbf{K})$ is the number of elements in set K.

Success Rate: Given a set of frames F, the success rate of a tracker is defined as the ratio of the number of successfully tracked frames and the total number of frames. The definition of a successfully tracked frame is the same as in the definition of average drift. Given index set K

$$SR(K, F) = \frac{\mathbf{card}(K)}{\mathbf{card}(F)} \tag{5}$$

We have compared our tracker (**RKLT**)(Ransac KLT) to 3 other trackers - the Inverse Compositional (**IC**) tracker [7], the ESM tracker (**ESM**) [2] and the Nearest Neighbor based tracker (**NNIC**) [3]. These have been chosen specifically because, to the best of our knowledge, these are the best 8 DOF trackers available. We used the trackers' implementation in [3] for the first three trackers as their source code is publicly available. Since parameter settings can have a significant impact on the performance of different trackers, we retain the original parameter settings as mentioned in the respective papers for all the three trackers to reasonably compare them with our tracker. For the IC tracker, we used template resolution R of 100×100 and maximum of 30 iterations. For the ESM tracker, the resolution used is 50×50 with the same number of maximum iterations. The NNIC tracker has the same configuration as in [3], with 3 layers of individual Nearest Neighbour trackers followed by an IC tracker.

While setting the parameters for our new tracker, we followed the same method that [3] used to set theirs, i.e. maximizing tracking performance while trying to give each algorithm approximately the same running time. So for our tracker, we used a template resolution of 40×40 and a maximum of 10 iterations for the IC tracker in the refinement step.

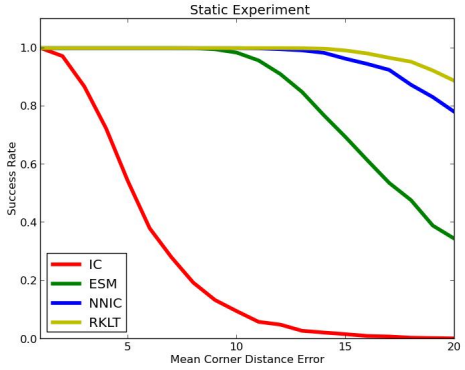

Figure 2. Static Experiment Result, success rate *TH* = 2

Table I
Challenges of sequences

Sequence	Object	Tracking Challenge
BookI	Book	Perspective Deformation(PD)
BookII	Book	Scaling
MugI	Coffee Mug	Low Texture
MugII	Coffee Mug	Low Texture, PD
MugIII	Coffee Mug	Low Texture, PD
Cereal	Cereal Box	Rotation

A. Static Experiments

The original static experiment proposed by [7] and later used by [2] [3] is designed like this: a subregion as region of interest with corner coordinates $c = \{c_1, c_2, c_3, c_4\}$, where $c_i \in \mathbb{R}^2$. is first selected in the original image I_0 (usually Lena's image). Next, a Gaussian noise with mean 0 and variance σ is added to each $c_i \in c$ and results in c'.

Based on c and c', a homography with parameters p can be estimated by using the DLT algorithm [20]. Given I_0, we can get a warped image $I_w = I_0(\mathbf{W}(p^{-1}, R))$ where $\mathbf{W}$ is the warp function and R is the set of reference points. The

Table II
AVERAGE DRIFT ON TMT DATASET

	IVT	L1	TLD	IC	ESM	NNIC	RKLT
BookI	1.93	3.04	N/A	0.62	0.62	0.68	0.70
BookII	3.61	2.27	N/A	0.58	0.45	0.46	0.50
MugI	3.20	1.82	3.18	0.36	0.41	0.37	0.45
MugII	2.30	1.78	3.43	1.70	1.70	1.71	1.70
MugIII	2.67	1.70	2.97	0.60	0.65	0.71	0.77
Cereal	N/A	1.97	N/A	0.07	0.22	0.22	0.40

N/A: No frame has MCD error below *TH=4* pixels

goal of this experiment is to initialize each tracker using I_0 with c and feed the warped image I_w to the tracker in order to estimate $\hat{c}$ which should be close to c'. In the complete experiment, the σ of Gaussian noise varies from 1 to 20, and 5000 trials are applied for each σ to estimate the SR (success rate) as defined in Eq. 5.

This standard test however does not give a precise indication of convergence. The issue is that, even for distributions with large σ, many samples are drawn with small motions. Hence a tracker will look as if it converges for a large perturbation σ, while most of successful trials in the graph may actually correspond to small motions. In contrast, we directly use MCD to represent the motion displacement. We vary MCD value α from 1 to 20 and for each α , we randomly generate a set P with 5000 set of warping parameters p_i. The $MCD(c_i, c')$ corresponding to each p_i in P is between $\alpha - 1$ and α. SR for each α is generated in the same way as in Eq. 5. The advantage of this modification is that it ensures that sufficient number of samples are generated for different magnitudes of motion displacement so that the result is statistically correct.

The results of this modified static experiment are summarized in Fig 2 which clearly shows each tracker's performance (in terms of SR) for different values of α. The success rate of IC tracker drops steeply for MCD value ≥ 1 while that of the ESM tracker begins to decrease at around 10. The NNIC and our new tracker outperform the other two with our tracker being slightly better.

B. Experiments on TMT dataset

In this experiment, we evaluated the four registration based trackers in addition to some learning based trackers on the TMT dataset [6], which is specifically designed for evaluating registration based trackers. We used both the evaluation metrics AD and SR introduced earlier for these experiments. [6] provides about 100 different image sequences. In order to evaluate our trackers with all motion types and with different target objects, we picked *bookI*, *bookII*, *mugI*, *mugII*, *mugIII* and *cereal* with high motion speeds s4 and s5. The challenges for each of these sequences can be found in Table I.

Average Drift Results : The average drift of each of the tested trackers can be found in Table II. In this experiment, we not only tested the registration based trackers used in the static experiments, but also several learning based trackers to show how the two classes of trackers compare. These latter include IVT [12], L1 [15] and TLD [13] trackers with their default parameter settings. A pixel threshold of 4 is used for computing AD. The results in Table II clearly show that, for most sequences, the AD for registration based trackers is much lower than that for the learning based trackers. Our tracker's results are similar to other registration based trackers.

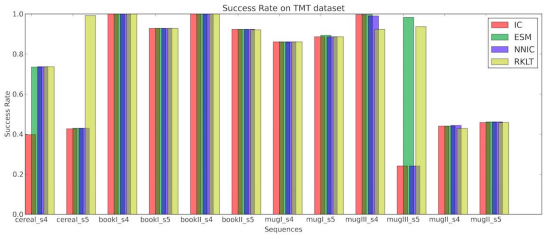

Figure 3. Success Rate Comparison among IC, ESM, NNIC and RKLT

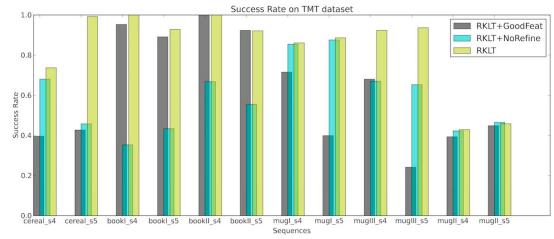

Figure 4. Success Rate Comparison among RKLT and two variants

Success Rate Results: The success rates for the registration based trackers are shown in Fig. 3. We can observe that our tracker is either the best or the second best in all the videos while in *cereal_s5*, it outperforms the others with almost twice the success rate. We can also observe that IC performs significantly worse than the others in two sequences and can generally be regarded as the worst of these trackers - an expected trade off for its higher speed. In *mugIII_s5*, ESM performs the best and is closely followed by our tracker. The fact that our tracker did not outperform the others in the two *mugIII* sequences, even though it was not far behind, may be due to the low textured appearance of the mug object which hampers RANSAC somewhat.

In order to clearly show the tracking progress, we have also plotted the MCD error of each frame on *cereal_s5* and *mugIII_s5* in Fig. 5. Tracked image examples can be found in Fig. 6

In addition to these four trackers, we also introduce another type of comparison to show the significance of the four steps in our approach. Two more trackers are implemented by making small modifications to our algorithm. The first one selects points using the feature detection

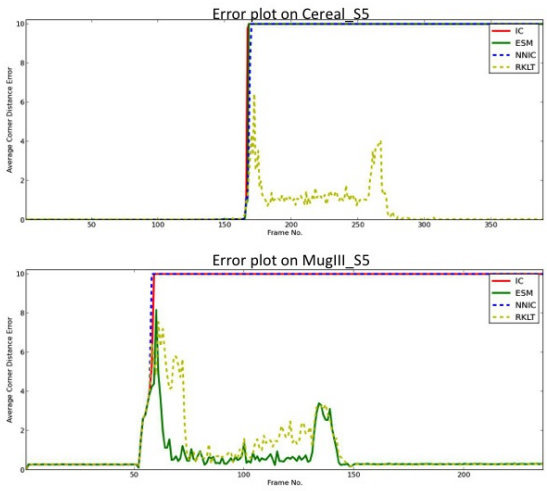

Figure 5. Top figure is the error plot for cereal fast speed sequence. Bottem figure is the error plot for *mugIII* fast speed sequence.

Table III
TRACKING EFFICIENCY ON TMT DATASET (FPS)

	IC	ESM	NNIC	RKLT
bookI	272.67	73.39	82.78	60.41
bookII	418.55	139.81	82.89	59.64
mugI	180.18	51.74	81.57	61.47
mugII	69.04	43.78	74.85	56.06
mugIII	164.21	62.01	76.79	64.02
Cereal	122.49	67.08	78.64	63.08

Figure 6. Tracking results of TMT dataset. From top to bottom: *cereal*, *mugII*, *mugIII*. Trackers include IC, ESM, NNIC, RKLT

method described in [16] instead of using evenly sampled points. We refer to it as **RKLT+GoodFeat** tracker. The other one removes the refinement step while leaving other steps unchanged and thus we refer to it as **RKLT+NoRefine** tracker. The results are shown in Fig. 4. RKLT+GoodFeat tracker can track well enough on *bookI* and *bookII*, but when there is image blur (*cereal*) or the object has low texture (*mugI*, *mugII*) it tracks poorly. RKLT+NoRefine tracks well only on *mugI* which only has translational motion but is significantly worse for all other sequences that have more complex motions. It can also be easily seen that our tracker using all four steps perform better than both RKLT+GoodFeat and RKLT+NoRefine on all the sequences.

C. Experiment on Metaio Dataset

Metaio's dataset is another popular dataset for registration based trackers released by Lieberknecht, et al [4] in 2009. The dataset includes 8 different target objects with each of them containing 5 challenges: angle, range, fast far, fast close, and illumination. The results of the four trackers on this dataset are summarized in Tables IV. The performance is measured in terms of the success rate for each sequence in percentage. For the Metaio benchmark we first counted the number of times each tracker got the highest number of frames tracked. IC was best 2 times, ESM 10 times, NNIC 16 times. Our proposed RKLT algorithm won 17 tests. We also observe that RKLT obviously surpasses the other two algorithms in *Angle* and *Fast Far* sequences. The former involves lots of in-plane/out-of-plane rotations, the latter contains relatively small tracking objects with fast motion. While in *Fast Close* where the objects become larger, RKLT performs not as good as ESM tracker. But for most of the sequences, our RKLT ranked in first two places. It is worth noting that the evaluation is performed at Metaio science detailed tracking ground truth coordinates are withheld from researchers.

D. Experiment on Time efficiency

Tracking speed may also affect the performance of a tracker in real world applications. For online tracking, the tracker needs to process the image feed from the camera immediately because the object may keep moving while the processing is going on. A slower tracker will automatically down sample the image frames (i.e. skip intermediate frames) and the object motion displacement between two adjacent frames it processes will consequently be larger than for a tracker with higher tracking speed. Hence there is a tradeoff between tracking accuracy and speed in real applications. Since the frame rates of most cameras does not exceed 30 Hz, we claim that if a tracker can track at a speed above 30 fps (frames per second) in a single thread on a normal research machine, it is good enough to be used in real applications.

In Table III we present the tracking time cost for all the trackers for each sequence while running the success rate experiment on the TMT dataset. We average the time cost for different speeds (s4/s5) of the same type of sequence and represent the tracking efficiency in fps. Note that we only take into account the time spent running the tracking algorithm in a single processing thread. The measurements

Table IV
METAIO DATASET RESULTS

Seq01	IC	ESM	NNIC	RKLT
Bump	78.3	*77.4*	68.1	77.2
Stop	100	100	100	100
Lucent	32	42.4	86.6	*68.8*
Board	22.6	*24.8*	10.1	77.6
Isetta	62.5	83.2	99.2	*97.5*
Philadelphia	63	82.1	*99.2*	100
Grass	11.4	16.6	*19.5*	47.6
Wall	51.7	64.1	*91*	98.9

Seq02	IC	ESM	NNIC	RKLT
Bump	70.9	91	*72.4*	*72.4*
Stop	72.8	96.2	*87.2*	83.8
Lucent	21.8	24.2	97.8	*51.7*
Board	7.9	*9.7*	2.8	20.2
Isetta	33.2	89	*83.1*	81.3
Philadelphia	53.2	89	99	*98.8*
Grass	8.6	*9.1*	25	7.2
Wall	26.1	39.3	98.2	*79.8*

Seq03	IC	ESM	NNIC	RKLT
Bump	20.6	50.8	56.5	*55.4*
Stop	12.2	24.9	*37.1*	68.2
Lucent	8.9	12.1	45.8	*21.9*
Board	4.6	*6.9*	5.7	10.4
Isetta	6.2	17.2	*74.8*	75.8
Philadelphia	6.7	14.6	38.2	*21.4*
Grass	5.7	7	7.8	*7.1*
Wall	8.1	8.4	70.9	*41.1*

Seq04	IC	ESM	NNIC	RKLT
Bump	30.6	40.9	33.8	*35.3*
Stop	30.4	54.1	*50.1*	26.9
Lucent	14.7	*17.6*	59.3	13.8
Board	23.5	*37.8*	3.9	38.1
Isetta	18	31.9	*21.4*	19.3
Philadelphia	50.5	*76.8*	78.2	58
Grass	3.7	*7.8*	5.4	8.4
Wall	12.4	*28.5*	24.2	50.4

Seq05	IC	ESM	NNIC	RKLT
Bump	*94.2*	95.9	78.8	48.2
Stop	62.7	62.9	75.8	*69.6*
Lucent	78.9	*89.6*	72.2	100
Board	11.9	*15.4*	0.8	39.4
Isetta	91.8	100	77.3	*99.5*
Philadelphia	99.4	100	100	100
Grass	14.2	*23.9*	8.9	31.7
Wall	73	*81.3*	72.9	82.2

Success rate of two challenges of different objects(in percentage). Seq01: Angle; Seq02:Range; Seq03:Fast Far; Seq04: Fast Close; Seq05:Illumination. Best tracker marked as red; Second best in *green*.

were done on a desktop computer with 3.5 GHz Intel processor and 32 GB RAM.

Our tracker is not the fastest but is reasonably fast. The speed is stable among all the different sequences and is close to 60 fps. IC runs the fastest in most sequences which is expected since it is the simplest and most efficient algorithm, followed by NNIC. ESM has similar performance as ours but with a larger variance.

V. CONCLUSION AND FUTURE WORK

In this paper we presented a novel registration based tracking algorithm that uses a combination of existing methods to yield state-of-the-art performance. First, several small patches are tracked by KLT trackers. Next a consistent homography motion is estimated using RANSAC. Finally, to achieve subpixel accuracy, an inverse compositional (IC) tracker is used to refine the result of the last step. The IC step also converges more robustly than in [1] for two reasons. Firstly, the RANSAC approximation puts it closer to the minimum, and thus more likely in the elliptical region of the error functional. Secondly, discarding pixels from the small KLT patches deemed as outliers by RANSAC removes data where the numerically similar KLT algorithm diverged on a small patch. These are likely to contain occlusions, specularities and other violations of the image constancy assumption in registration tracking. Since IC optimized in the least squares sense, removing these high residual outliers significantly increases convergence.

Since the result of RANSAC serves as a good initial guess, the IC algorithm converges significantly faster than IC and other similar registration trackers used alone. We also showed, through 3 sets of experiments, that our tracker performs comparably to the best in high DOF tracking.

For future work, although we briefly discussed the difference between feature detection/matching algorithms and ours, we did not show the difference experimentally. A good comparison both in theory and through experiments would be interesting. Another avenue for improvement is our algorithm's failure detection part that is somewhat ad-hoc at present. A better idea can be to apply some similarity measure between the template and the tracking result using criteria like NCC(Normalized Cross Correlation) which may make the tracker more robust.

REFERENCES

[1] Baker, Simon, and Iain Matthews, "Lucas-kanade 20 years on: A unifying framework," *International journal of computer vision*, 56.3 (2004): 221-255.

[2] Benhimane, Selim, and Ezio Malis. "Real-time image-based tracking of planes using efficient second-order minimization." In *IROS 2004*. vol. 1, pp. 943-948.

[3] Dick, Travis, et al. "Realtime Registration-Based Tracking via Approximate Nearest Neighbour Search." *Robotics: Science and Systems*. 2013.

[4] Lieberknecht, Sebastian, et al. "A dataset and evaluation methodology for template-based tracking algorithms." In *ISMAR*, 2009.

[5] Comport, Andrew I., et al. "Real-time markerless tracking for augmented reality: the virtual visual servoing framework." *Visualization and Computer Graphics*, IEEE Transactions on, 12.4 (2006): 615-628.

[6] Roy, Ankush, Xi Zhang, et al. "Tracking Benchmark and Evaluation for Manipulation Tasks." In *ICRA*, 2015.

[7] Baker, Simon, and Iain Matthews. "Equivalence and efficiency of image alignment algorithms." In *CVPR*, 2001.

[8] Bouguet, Jean-Yves. "Pyramidal implementation of the lucas kanade feature tracker description of the algorithm." *Intel Corporation*, 5, 2001, 1-10.

[9] Jurie, Frdric, and Michel Dhome. "Hyperplane approximation for template matching." *PAMI*, 24.7 (2002): 996-1000.

[10] Marius Muja and David G. Lowe, "Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration", *VISAPP*, 2009

[11] Wu, Yi, Jongwoo Lim, and Ming-Hsuan Yang. "Online object tracking: A benchmark." *CVPR*, 2013

[12] Ross, David A., et al. "Incremental learning for robust visual tracking." *IJCV*, 77.1-3 (2008): 125-141.

[13] Kalal, Zdenek, Krystian Mikolajczyk, and Jiri Matas. "Tracking-learning-detection." *PAMI*, 34.7 (2012): 1409-1422.

[14] Zhang, Kaihua, Lei Zhang, and Ming-Hsuan Yang. "Real-time compressive tracking." *ECCV* 2012. 864-877.

[15] Bao, Chenglong, et al. "Real time robust l1 tracker using accelerated proximal gradient approach." *CVPR* 2012.

[16] Shi, Jianbo, and Carlo Tomasi. "Good features to track." *CVPR*, 1994.

[17] Benhimane, Selim, et al. "Linear and quadratic subsets for template-based tracking." *CVPR*, 2007.

[18] Kalal, Zdenek, Krystian Mikolajczyk, and Jiri Matas. "Forward-backward error: Automatic detection of tracking failures." *ICPR*, 2010.

[19] Brown, Matthew, and David G. Lowe. "Automatic panoramic image stitching using invariant features." *IJCV*, 74.1 (2007): 59-73.

[20] Richard Hartley and Andrew Zisserman, "Multiple View Geometry in computer vision." *Cambridge University Press*, 2003.

[21] Fischler, Martin A., and Robert C. Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography." *Communications of the ACM*, 1981.

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