

Real-time Model-based Rigid Object Pose Estimation and Tracking Combining Dense and Sparse Visual Cues

Karl Pauwels Leonardo Rubio Javier Díaz Eduardo Ros
University of Granada, Spain
{kpauwels, lrubio, jda, eros}@ugr.es

Abstract

We propose a novel model-based method for estimating and tracking the six-degrees-of-freedom (6DOF) pose of rigid objects of arbitrary shapes in real-time. By combining dense motion and stereo cues with sparse keypoint correspondences, and by feeding back information from the model to the cue extraction level, the method is both highly accurate and robust to noise and occlusions. A tight integration of the graphical and computational capability of Graphics Processing Units (GPUs) results in pose updates at framerates exceeding 60 Hz. Since a benchmark dataset that enables the evaluation of stereo-vision-based pose estimators in complex scenarios is currently missing in the literature, we have introduced a novel synthetic benchmark dataset with varying objects, background motion, noise and occlusions. Using this dataset and a novel evaluation methodology, we show that the proposed method greatly outperforms state-of-the-art methods. Finally, we demonstrate excellent performance on challenging real-world sequences involving object manipulation.

1. Introduction

Estimating and tracking the 6DOF (three translation and three rotation) pose of rigid objects is critical for robotic applications involving object grasping and manipulation, and also for activity interpretation. In many situations, models of the object of interest can be obtained off-line, or on-line in an exploratory stage. Since it is such an important ability of robotic systems, a wide variety of methods have been proposed in the past. We only provide a brief overview of the major classes of methods here. Many more examples exist for each class.

The most popular tracking methods match expected to observed edges by projecting a 3D wireframe model in the image [6]. Many extensions have been proposed that exploit also texture information and particle filtering in order to reduce sensitivity to background clutter and noise [5, 13].

Tracking-by-detection approaches on the other hand can recover the pose without requiring an initial estimate by using sparse keypoints and descriptors with wide-baseline matching [9, 13]. These are related to techniques for multi-view object class detection, but the latter typically provide only coarse viewpoint estimates [10]. A different class of approaches relies on level-set methods to maximize the discrimination between statistical foreground and background appearance models [17]. These methods can include additional cues such as optical flow and sparse keypoints, but at a large computational cost [4]. Most of the above-mentioned approaches can exploit multi-view information, but dense depth information is rarely used. Recently, due to the prevalence of cheap depth sensors, depth information is being applied with great success in various related problems, such as on-line scene modeling [15], articulated body pose estimation [18], and texture-less object detection [7]. We show here that incorporating these advances in real-time object tracking, and extending them with additional cues, yields great improvements.

The main contributions can be summarized as follows. First of all, we introduce a novel model-based 6DOF pose estimation and tracking method that combines dense motion and stereo measurements with sparse keypoint features. It exploits feedback from the model to the cue extraction level and provides an effective measure of reliability. Secondly, the method has been designed specifically for high-performance operation (> 60 Hz) and this is achieved through the exploitation and tight integration of the graphics and computation pipelines of modern GPUs in both the low-level cue extraction and pose estimation stages. Finally, an extensive benchmarking dataset and evaluation methodology have been developed and used to show increased performance (in accuracy, robustness, and speed) as compared to state-of-the-art methods.

2. Proposed Method

A concise overview of the method is shown in Fig. 1. Different visual cues are extracted from a stereo video

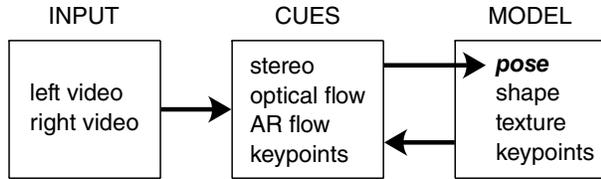


Figure 1. Method overview illustrating the different visual cues and model components (AR = Augmented Reality, see 2.2.2). The cues are combined to estimate the object pose and model information is fed back to facilitate the cue extraction.

stream and combined with model information (known a priori) to estimate the 6DOF pose. In turn, model information is fed back to facilitate the cue extraction itself.

2.1. Model representation

The proposed method supports 3D models of arbitrary shape and appearance of the object to track. The model’s surface geometry and appearance are represented by a triangle mesh and color texture respectively. For a given pose, the color, distance to the camera, and surface normal at each pixel can be obtained efficiently through OpenGL. In a training stage, SIFT (Scale-Invariant Feature Transform) features [12] are extracted from keyframes of rotated versions of the model (30° separation), and mapped to the surface model.

2.2. Visual cues

The dense motion and stereo cues are obtained using modified coarse-to-fine GPU-accelerated phase-based algorithms [16], and the SIFT features are extracted and matched using a GPU library [20].

2.2.1 Model-based dense stereo

Coarse-to-fine stereo algorithms, although highly efficient, support only a limited disparity range and have difficulties detecting fine structures. To overcome these problems we feed the object pose estimate obtained in the previous frame back as a prior in the stereo algorithm. Figure 2A,B show an example real-world stereo image pair of a box being manipulated. Using the box’s previous frame pose estimate, stereo priors are generated for the current frame (Fig. 2C,D) and introduced at the lowest scale used by the stereo algorithm. The reliable estimates (after left-right consistency checking) for the current frame obtained with and without using the priors are shown in panels E and F respectively. Without the prior, the focus is more on the background than on the object of interest.

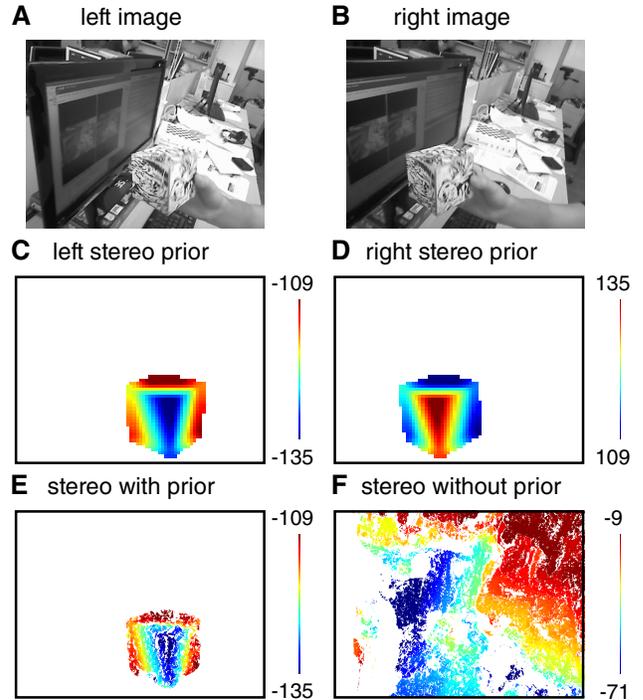


Figure 2. (A) Left and (B) right input images and low-resolution (C) left and (D) right stereo priors generated using the previous frame pose estimate of the manipulated box. (E) Stereo obtained using four scales and initializing with the prior, and (F) stereo obtained using six scales and without using the prior.

2.2.2 Dense motion cues

The optical flow algorithm integrates the temporal phase gradient across different orientations and also uses a coarse-to-fine scheme to increase the dynamic range. Unlike [16], we use only two frames, I_t and I_{t+1} , at times t and $t + 1$. A simple forward/backward consistency check discards unreliable estimates. A prior is not required since displacements in the motion scenario are much smaller than in the stereo scenario.

Figure 3C contains the (subsamped and scaled) optical flow vectors from Fig. 3A to the next image in a complex real-world scenario with both object and camera motion. We also extract a second type of motion which we refer to as Augmented Reality (AR) flow that includes feedback from the model. The model’s texture is rendered at the current pose estimate and overlaid on I_t , resulting in an ‘augmented image’ \hat{I}_t . An example is shown in Fig. 3B. The motion is then computed from \hat{I}_t to the next real image (I_{t+1}), and shown in Fig. 3D for the example presented here. Because of the erroneous pose estimate (deliberately large in this case to better illustrate the concept) this motion is quite different from the optical flow. It allows the tracker to recover from such errors by effectively countering the drift that results from tracking based on optical flow alone.

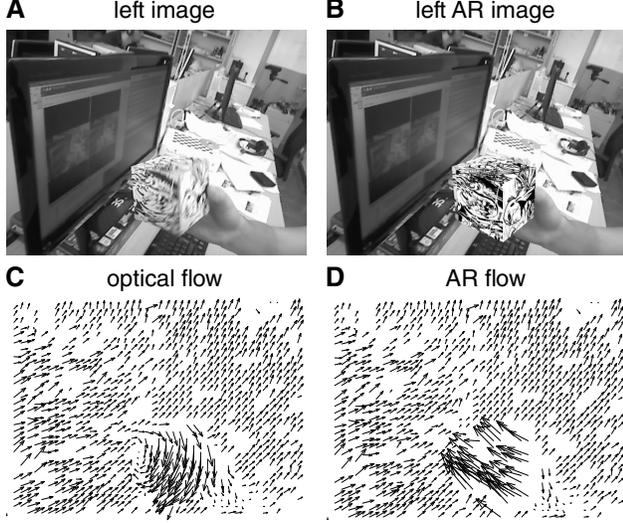


Figure 3. (A) Left image and (B) left image with rendered model (at incorrect pose) superimposed. (C) Real optical flow from real image in A to the next real image. (D) AR flow from AR image in B to the next real image. Both flow fields are subsampled ($15\times$) and scaled ($5\times$).

2.3. Pose Estimation and Tracking

The proposed method incorporates the differential rigid motion constraint into a fast variant of the iterative closest point (ICP) algorithm [1] to allow all the dense cues to simultaneously and robustly minimize the pose error. A selection procedure (2.3.2) is used to re-initialize the tracker with an estimate obtained from a sparse tracking-by-detection approach, if required.

2.3.1 Dense Pose Tracking

Our aim is to recover the rigid rotation and translation that best explains the dense visual cues and transforms each model point $\mathbf{m} = [m_x, m_y, m_z]^T$ at time t into point \mathbf{m}' at time $t + 1$:

$$\mathbf{m}' = \mathbf{R} \mathbf{m} + \mathbf{t}, \quad (1)$$

with \mathbf{R} the rotation matrix and $\mathbf{t} = [t_x, t_y, t_z]^T$ the translation vector. The rotation matrix can be simplified using a small angle approximation:

$$\mathbf{m}' \approx (\mathbf{1} + [\boldsymbol{\omega}]_{\times}) \mathbf{m} + \mathbf{t}, \quad (2)$$

with $\boldsymbol{\omega} = [\omega_x, \omega_y, \omega_z]^T$ representing the rotation axis and angle. Each stereo disparity measurement, d' , at time $t + 1$ can be used to reconstruct an approximation, \mathbf{s}' , to \mathbf{m}' :

$$\mathbf{s}' = \begin{bmatrix} s'_x \\ s'_y \\ s'_z \end{bmatrix} = \begin{bmatrix} x s'_z / f \\ y s'_z / f \\ -f b / d' \end{bmatrix}, \quad (3)$$

with $\mathbf{x} = [x, y]^T$ the pixel coordinates (with nodal point as origin), f the focal length, and b the baseline of the stereo rig. To use this reconstruction in (1), the model point \mathbf{m} corresponding to \mathbf{s}' needs to be found. We deliberately avoid using the motion cues to facilitate this correspondence finding, as done in many ICP extensions, since this requires both valid stereo and valid motion measurements at the same pixel. Instead we use the efficient projective data association algorithm [2] that corresponds stereo measurements to model points that project to the same pixel. Since these correspondences are less accurate, a *point-to-plane* as opposed to a *point-to-point* distance needs to be minimized [21], which is obtained by projecting the error on $\mathbf{n} = [n_x, n_y, n_z]^T$, the model normal vector in \mathbf{m} :

$$e = [(\mathbf{R} \mathbf{m} + \mathbf{t} - \mathbf{s}') \cdot \mathbf{n}]^2. \quad (4)$$

This error expresses the distance from the reconstructed points to the plane tangent to the model. By linearizing this measure as in (2) and summing over all correspondences $\{\mathbf{m}_i, \mathbf{s}'_i\}$, we arrive at a strictly shape-based error measure that is linear in the parameters of interest \mathbf{t} and $\boldsymbol{\omega}$:

$$e_S(\mathbf{t}, \boldsymbol{\omega}) = \sum_i \left([(\mathbf{1} + [\boldsymbol{\omega}]_{\times}) \mathbf{m}_i + \mathbf{t} - \mathbf{s}'_i] \cdot \mathbf{n}_i \right)^2. \quad (5)$$

By rearranging (2), we obtain the following:

$$\mathbf{m}' - \mathbf{m} \approx \mathbf{t} + \boldsymbol{\omega} \times \mathbf{m}. \quad (6)$$

Note that this looks very similar to the differential motion equation from classical kinematics that expresses the 3D motion of a point, $\dot{\mathbf{m}}$, in terms of 3D translational and rotational velocity:

$$\dot{\mathbf{m}} = \mathbf{t} + \boldsymbol{\omega} \times \mathbf{m}. \quad (7)$$

We deliberately retain the $(\mathbf{t}, \boldsymbol{\omega})$ notation since the displacements in (2) are expressed in the same time unit. This can now be used with the optical flow to provide additional constraints on the rigid motion. Unlike in the stereo case (3), $\dot{\mathbf{m}}$ cannot be reconstructed from the optical flow and instead (7) needs to be enforced in the image domain. After projecting the model shape:

$$\mathbf{x} = f \begin{bmatrix} m_x / m_z \\ m_y / m_z \end{bmatrix}, \quad (8)$$

the expected pixel motion $\dot{\mathbf{x}} = [\dot{x}, \dot{y}]^T$ becomes:

$$\dot{\mathbf{x}} = \frac{\delta \mathbf{x}}{\delta t} = \frac{f}{m_z^2} \begin{bmatrix} \dot{m}_x m_z - m_x \dot{m}_z \\ \dot{m}_y m_z - m_y \dot{m}_z \end{bmatrix}. \quad (9)$$

Combining (9) with (7) results in the familiar equations [11]:

$$\dot{x} = \frac{(f t_x - x t_z)}{m_z} - \frac{x y}{f} \omega_x + (f + \frac{x^2}{f}) \omega_y - y \omega_z, \quad (10)$$

$$\dot{y} = \frac{(f t_y - y t_z)}{m_z} - (f + \frac{y^2}{f}) \omega_x + \frac{x y}{f} \omega_y + x \omega_z, \quad (11)$$

which are linear in \mathbf{t} and $\boldsymbol{\omega}$ provided the depth of the point is known. We obtain this depth m_z by rendering the model at the current pose estimate (8). Since we have two sources of pixel motion, we have two error functions:

$$e_O(\mathbf{t}, \boldsymbol{\omega}) = \sum_i \|\dot{\mathbf{x}}_i - \mathbf{o}_i\|^2, \quad (12)$$

$$e_A(\mathbf{t}, \boldsymbol{\omega}) = \sum_i \|\dot{\mathbf{x}}_i - \mathbf{a}_i\|^2, \quad (13)$$

with $\mathbf{o} = [o_x, o_y]^T$ and $\mathbf{a} = [a_x, a_y]^T$ the observed optical and AR flow respectively.

Both the linearized point-to-plane distance in the stereo case and the differential motion constraint in the optical and AR flow case now provide linear constraints on the same rigid motion representation $(\mathbf{t}, \boldsymbol{\omega})$ and can thus be minimized jointly using the following error function:

$$E(\mathbf{t}, \boldsymbol{\omega}) = e_S(\mathbf{t}, \boldsymbol{\omega}) + e_O(\mathbf{t}, \boldsymbol{\omega}) + e_A(\mathbf{t}, \boldsymbol{\omega}). \quad (14)$$

To increase the robustness, an M-estimation scheme is used to gradually reduce the influence of and remove outliers from the estimation [14]. In the case of large rotations these linearized constraints are only crude approximations, and many of the shape correspondences obtained through projective data association will be wrong. Therefore, the minimization of (14) needs to be iterated a number of times, at each iteration updating the pose, the shape correspondences, and the unexplained part of the optical and AR flow measurements.

At iteration k , an incremental pose update is obtained by minimizing $E(\Delta\mathbf{t}^k, \Delta\boldsymbol{\omega}^k)$, and accumulated into the pose estimate at the previous iteration $k-1$:

$$\mathbf{R}^k = \Delta\mathbf{R}^k \mathbf{R}^{k-1}, \quad (15)$$

$$\mathbf{t}^k = \Delta\mathbf{R}^k \mathbf{t}^{k-1} + \Delta\mathbf{t}^k, \quad (16)$$

where $\Delta\mathbf{R}^k = e^{[\Delta\boldsymbol{\omega}^k]_\times}$. The model is updated as:

$$\mathbf{m}^k = \mathbf{R}^k \mathbf{m} + \mathbf{t}^k, \quad (17)$$

and used to obtain the new (projective) shape correspondences and the part of the optical and AR flow explained thus far, $\Delta\mathbf{x}^k = [\Delta x^k, \Delta y^k]^T$:

$$\Delta x^k = f(m_x^k/m_z^k - m_x/m_z), \quad (18)$$

$$\Delta y^k = f(m_y^k/m_z^k - m_y/m_z). \quad (19)$$

This explained flow is subtracted from the observed optical and AR flow:

$$\mathbf{o}^k = \mathbf{o} - \Delta\mathbf{x}^k, \quad (20)$$

$$\mathbf{a}^k = \mathbf{a} - \Delta\mathbf{x}^k. \quad (21)$$

The next iteration incremental pose updates are then obtained by minimizing $E(\Delta\mathbf{t}^{k+1}, \Delta\boldsymbol{\omega}^{k+1})$, which operates

on \mathbf{o}^k , \mathbf{a}^k , \mathbf{m}^k , and s' . This cycle is repeated a fixed number of times (we use three internal iterations for the M-estimation, and three external iterations). Note that even though the updates in each iteration are estimated from linearized equations, the correct accumulated discrete updates are used in between iterations (17–19).

2.3.2 Combined Sparse and Dense Pose Estimation

A RANSAC-based monocular perspective-n-point pose estimator is used to robustly extract the 6DOF object pose on the basis of correspondences between image (2D) and model codebook (3D) SIFT keypoint descriptors [3, 9]. Exhaustive matching is used and therefore, unlike the dense component of the proposed method, this sparse estimator provides a pose estimate that does not depend on the previous frame's estimate.

Due to the nonlinear and multimodal nature of the sparse and dense components, directly merging them using for instance a Kalman filtering framework is not suitable here. Instead we *select* either the sparse or dense estimate based on the effectiveness of its feedback on cue extraction, as measured by the proportion of valid AR flow vectors in the projected object region. A flow vector is considered valid if it passes a simple forward/backward consistency check (2.2.2). This measure is used since, unlike dense optical-flow-based or stereo-based measures, AR flow is affected by object occlusions. We will show in Section 4 that this simple measure is adequate both for selecting and determining the reliability of the pose estimate.

2.4. Processing Times

A breakdown of the time required to process a single 640×480 image frame is provided in Table 1 (using one core of a Geforce GTX 590). The AR flow is computed twice, based on the previous frame's sparse and dense estimate. Consequently, four Gabor pyramids need to be computed (left, right, $2 \times$ left AR). A number of rendering steps are also required to create the stereo priors and AR images. Note that the low-level component computes dense stereo and three times optical flow altogether at ± 95 Hz. Dense pose estimation takes 4.8 ms for a configuration with 50,000 data samples, three internal (M-estimation) and three external (ICP) iterations. These timings scale very well with model complexity. The total time spent on rendering is 1.65 ms for the 800 triangle model used in Table 1, 1.78 ms for a 25,000 triangle model, and 2.6 ms for a 122,000 triangle model. The tracking module requires 16 ms in total and thus exceeds 60 Hz.

The sparse component runs independently on the second GPU core so that it does not affect the tracker's speed. Its estimates are employed when available and so its speed mainly determines the recovery latency in case the dense-

Table 1. Processing times dense tracking (in ms)

dense cues (640×480 image size)		10.6
– Gabor pyramid ($4 \times$)	3.8	
– optical flow	1.5	
– AR flow ($2 \times$)	3.2	
– model-based stereo	1.3	
– rendering	0.8	
dense pose ($50,000$ samples)		4.8
– residual flow (18–21)	0.5	
– rendering	0.8	
– gathering/compaction	1.4	
– pose estimate	2.1	
error evaluation		0.6
total		16.0



Figure 4. Object models used.

based tracker is lost. Our current implementation runs at 20 Hz. This can be increased by reducing the model size and/or using more efficient keypoints and descriptors, but at the cost of reduced accuracy.

3. Synthetic Benchmark Dataset

We have constructed an extensive synthetic benchmark dataset to evaluate pose trackers under a variety of realistic conditions. Its creation is discussed in detail next, but Fig. 7 already shows some representative examples that illustrate the large distance range, background and object variability, and the added noise and occlusions. The complete dataset is available on-line.¹

3.1. Object Models

We have selected four objects from the publicly available KIT object models database [8]. Snapshots of these models are shown in Fig. 4 (*soda*, *soup*, *clown*, and *candy*) and provide a good sample of the range of shapes and textures available in the database. We also included two cube-shaped objects, one richly textured (*cube*), and the other containing only edges (*edge*).

3.2. Object and Background Motion

Using the proposed system, we recorded a realistic and complex motion trace by manually manipulating an object

¹<http://www.karlpauwels.com>

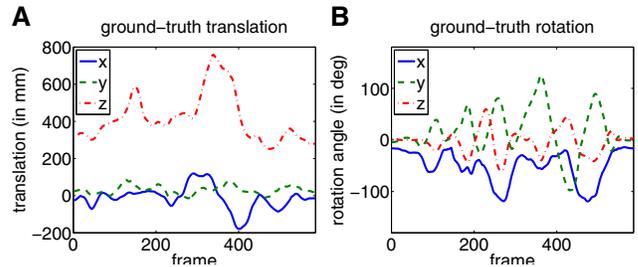


Figure 5. Ground-truth object motion in synthetic sequences.

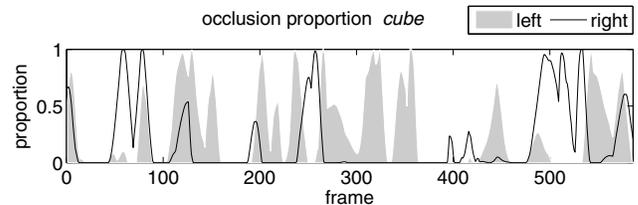


Figure 6. Proportion occlusion in the *cube* sequence.

similar to *cube*. This, possibly erroneous, motion trace was then used to generate synthetic sequences and so it is, by definition, the ground-truth object motion. The trace is shown in Fig. 5 and covers a high dynamic range, varying all six degrees of freedom. The realism and complexity of the sequences is further increased by blending the rendered objects into a real-world stereo sequence recorded with a moving camera in a cluttered office environment. Some examples are shown in Fig. 7 but, to fully appreciate the complexity, we refer to the supplemental material video.

3.3. Noise and Occluder

To further explore the limitations of pose estimation methods, different sequences are created corrupted either by noise or an occluding object. For the noisy sequences, Gaussian noise ($\sigma = 0.1$ intensity) is added separately to each color channel, frame, and stereo image (Fig. 7B). To obtain realistic occlusion (with meaningful motion and stereo cues), we added a randomly bouncing 3D teddy bear object to the sequence (Fig. 7C,D and the supplemental material). The occlusion proportion of the *cube* object over the sequence is shown in Fig. 6. Although this differs for the left and right sequences, none of the methods evaluated here exploit this (*e.g.* our dense stereo cue is affected by *either* left or right occlusions).

3.4. Performance Evaluation

Pose trackers are usually evaluated by comparing the estimated to the ground-truth or approximate ground-truth pose across an entire sequence [4, 5, 17]. However, once a tracker is lost, the subsequent estimates (and their errors) become irrelevant. For example, if tracking is lost early in the scene, the average error will typically be very large, but

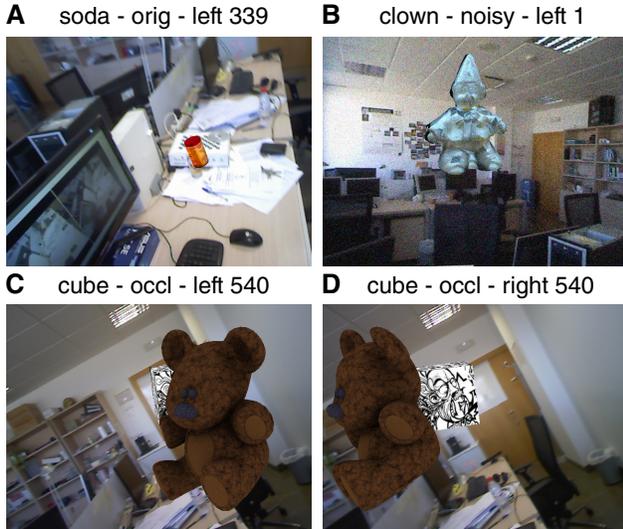


Figure 7. Indicative samples from the different synthetic sequences illustrating (A) large distance range, (B) added noise, and (C,D) different occlusion proportions in the left and right sequences (orig = noise free ; occl = occluded ; number = frame index).

this doesn't mean the tracker cannot track the object in the remainder of the sequence, if re-initialized.

For this reason, we propose to instead measure the *proportion* of a sequence that can be tracked successfully. Since we use synthetic sequences, we can continuously monitor tracking accuracy and automatically reset when tracking is lost. But how to decide if tracking is lost? Rotations and translations affect an object's position in very different ways and are therefore hard to summarize into a single error. Instead we use the largest distance between corresponding vertices, \mathbf{v}_j , of the object transformed according to the ground-truth (\mathbf{R}, \mathbf{t}) and estimated pose $(\hat{\mathbf{R}}, \hat{\mathbf{t}})$:

$$e_P = \max_j \|(\hat{\mathbf{R}} \mathbf{v}_j + \hat{\mathbf{t}}) - (\mathbf{R} \mathbf{v}_j + \mathbf{t})\|. \quad (22)$$

When this distance exceeds a threshold (e.g. 10 mm), the tracker is reset to the ground-truth. The proportion of the sequence tracked correctly then constitutes a scalar performance measure for the entire sequence. To put this measure in perspective, a *static* error is also computed for each sequence using a 'naive tracker'. This 'tracker' simply never updates the pose. As a consequence all resets are triggered by the object motion only and the error provides an indication of the sequence complexity. For example, in a sequence with a static object, perfect performance will be achieved.

4. Results

4.1. Stereo and Optical Flow Synergy

Table 2 shows results on the least textured sequences *soda* and *edge*. The *static* performance is around 50% in

Table 2. Tracking success rate (in %) – stereo and optical flow

	<i>soda</i>			<i>edge</i>		
	orig	noisy	occl	orig	noisy	occl
static	53	53	53	50	50	50
stereo	77	47	42	59	50	32
optical flow	93	81	57	78	81	40
stereo+flow	100	96	63	92	93	52

both which means that, without tracking, a reset is required approximately every other frame. Due to the low texture, shape-symmetry (*soda*) and shape-planarity (*edge*), stereo-only performance is quite bad in these sequences and even below *static* in the noisy and occluded scenarios. Optical-flow-only performance is better but, when both are combined, great improvements can be observed. This highlights the importance of combining multiple cues, particularly in low-texture situations. The AR flow and sparse cues further improve the results, but these are discussed in the next section (and shown in Table 3).

4.2. State-of-the-art Methods

The Blocks World Robotic Vision Toolbox (BLORT) [13] provides a number of object learning, recognition, and tracking components that can be combined to robustly track object pose in real-time. We only evaluate the particle-filter-based tracking module here since the recognition module is very similar to our sparse-only method. Each particle represents a pose estimate and is used to render an edge model (combining edges from geometry and texture) that is matched against edges extracted from the current image. We evaluate BLORT's tracking module with the default (real-time) setting with 200 particles and a high precision variant with 10,000 particles. Due to an inefficient rendering procedure, the current tracker implementation of BLORT can not handle models with a high vertex count. We therefore limited the geometrical complexity of the models to 800 triangles in all the sequences.

We also evaluate a state-of-the-art real-time-capable region-based tracker. The PWP3D method [17] uses a 3D geometry model to maximize the discrimination between statistical foreground and background appearance models, by directly operating on the 3D pose parameters. To ensure the best possible performance, we used very small gradient descent step sizes (0.1 degrees for rotation and 1 mm for translation). Together with a large number of iterations (100), this ensures stable convergence (although no longer in real-time). Furthermore, we initialized the PWP3D method at each frame with the ground-truth color histogram of the actual (or unoccluded) frame being processed so that also inaccuracies here do not affect performance.

Table 3 summarizes all the results obtained with a track-

Table 3. Tracking success rate (in %) – orig = noise free; occl = occluded

	<i>soda</i>			<i>soup</i>			<i>clown</i>			<i>candy</i>			<i>cube</i>			<i>edge</i>		
	orig	noisy	occl	orig	noisy	occl	orig	noisy	occl	orig	noisy	occl	orig	noisy	occl	orig	noisy	occl
sparse-and-dense	99	97	68	98	99	80	100	98	77	100	100	81	100	100	76	98	98	57
dense-only	100	98	67	100	99	74	100	100	70	100	100	75	100	100	71	98	99	57
sparse-only	61	37	44	93	74	77	92	71	74	96	91	80	98	96	79	0	0	0
part. filt. 10,000	76	65	54	77	66	63	88	82	76	77	76	64	93	94	76	72	91	68
region-based	84	84	44	96	96	44	96	89	44	84	84	39	84	74	38	85	84	39
part. filt. 200	58	60	45	47	54	40	56	62	48	46	49	41	53	54	39	63	63	50
static	53	53	53	45	45	45	47	47	47	46	46	46	50	50	50	50	50	50

ing reset threshold equal to 10 mm. The proposed *dense-only* tracking method obtains an almost perfect performance regardless of model shape, texture (see *edge*), or sequence noise. In the occluded scenario however, it is frequently outperformed by the *sparse-only* and high quality *particle filtering* methods. But, when combined with the sparse method (2.3.2) the synergy of both modules is confirmed. *Sparse-and-dense* retains the excellent performance of the *dense-only* method with greatly improved robustness to occlusions, even outperforming *sparse-only* on most sequences. Note that, unlike *dense-only*, *sparse-and-dense* also enables recovery from tracking failures. The *particle filter* method performs very well provided a very large number of particles are used. In the real-time setting (200 particles) the performance is not much better than *static*. The *sparse-only* method performs badly on *soda* due to the weak texture, and fails on *edge* due to the complete absence of texture. The *region-based* tracker performs consistently well on the noise-free and noisy sequences, but fails dramatically in the presence of an occluder. Although it can handle certain types of occlusions, large failures occur when an entire side of the object contour is occluded [17].

4.3. Real-world Sequences

The proposed method also performs excellently in real-world scenarios. Some example results with a cluttered scene, occluders, and camera motion are shown in Fig. 8. The dense estimate is selected as winner in Fig. 8A,B. This usually occurs when the object is far away (A) or suffering from severe motion blur (B). The sparse estimate is usually selected when only a small part of the object is visible (C). Figure 8D finally shows some tracking failures that are detected correctly by the reliability measure (proportion AR flow < 0.15). See the supplemental material video for more real-world results.

5. Discussion

Although highly robust, the edge-based particle filter method requires a large number of particles to achieve high accuracy. Since each particle requires a rendering step,

the performance critically depends on model complexity. This method is also difficult to extend to the articulated scenario due to the increased dimensionality of the problem. The sparse keypoint-based method is highly robust to occlusions and provides excellent synergy with the dense methods proposed here. The region-based method does not require edges or texture and performs very well. It does have problems with symmetric objects, is slow to converge, and fails on certain types of occlusions. This requires the use of multiple cameras or explicit modeling of the occlusions. Note that the proposed method also supports depth cues other than stereo (e.g. from a Kinect sensor), and conversely, enables for the incorporation of motion cues in current depth-only applications [15]. Current Kinect versions however do not provide the high shape detail close to the camera, nor the high framerates achieved by our model-based stereo algorithm [19]. Finally, the articulated case represents a relatively straightforward extension of the proposed methods, and the high-level of detail provided by the dense cues should be especially powerful in this, and also the deformable, scenario.

6. Conclusion

We have presented a novel multi-cue approach for the 6DOF model-based pose tracking of rigid objects of arbitrary shapes that exploits dense motion and stereo cues, sparse keypoint features, and feedback from the model to the cue extraction. The method is inherently parallel and efficiently implemented using GPU acceleration. We have introduced an evaluation methodology and benchmark dataset specifically for this problem. Using this dataset we have shown improved accuracy, robustness, and speed of the proposed method as compared to state-of-the-art real-time-capable methods.

Acknowledgments

This work is supported by grants from the European Commission (Marie Curie FP7-PEOPLE-2011-IEF-301144 and TOMSY FP7-270436) and CEI BioTIC GENIL (PYR-2012-9).

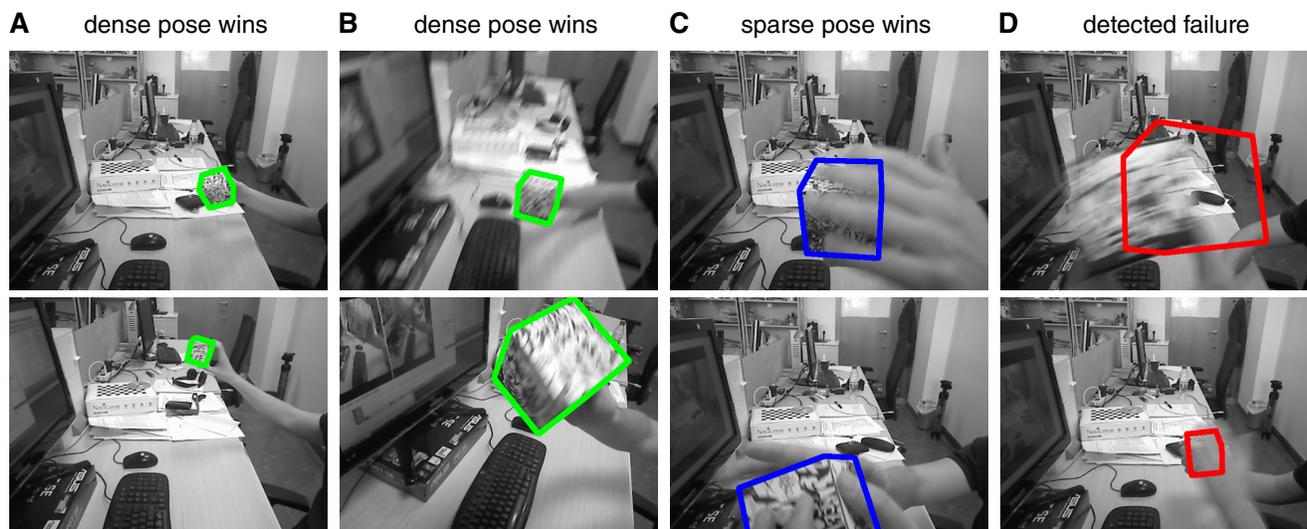


Figure 8. Indicative real-world pose estimation results, showing how the dense pose is selected when (A) the object is far and/or (B) motion-blurred, how the sparse pose is selected in case of (C) strong occlusions, and how (D) failures can be detected correctly.

References

- [1] P. Besl and N. McKay. A method for registration of 3D shapes. *IEEE PAMI*, 14:239–256, 1992. 3
- [2] G. Blais and M. Levine. Registering multiview range data to create 3D computer objects. *IEEE PAMI*, 17(8):820–824, 1995. 3
- [3] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000. 4
- [4] T. Brox, B. Rosenhahn, J. Gall, and D. Cremers. Combined region and motion-based 3D tracking of rigid and articulated objects. *IEEE PAMI*, 32(3):402–415, 2010. 1, 5
- [5] C. Choi and H. I. Christensen. Robust 3D visual tracking using particle filtering on the special euclidean group: A combined approach of keypoint and edge features. *Int. J. Robot. Res.*, 31(4):498–519, 2012. 1, 5
- [6] T. Drummond and R. Cipolla. Real-time visual tracking of complex structures. *IEEE PAMI*, 24(7):932–946, 2002. 1
- [7] S. Hinterstoisser, S. Holzer, C. Cagniard, S. Ilic, K. Konolige, N. Navab, and V. Lepetit. Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In *ICCV*, pages 858–865, 2011. 1
- [8] A. Kasper, Z. Xue, and R. Dillmann. The KIT object models database: An object model database for object recognition, localization and manipulation in service robotics. *Int. J. Robot. Res.*, 31(8):927–934, 2012. 5
- [9] V. Lepetit and P. Fua. Monocular model-based 3D tracking of rigid objects. *Foundations and Trends in Computer Graphics and Vision*, 1:1–89, 2005. 1, 4
- [10] J. Liebelt and C. Schmid. Multi-view object class detection with a 3D geometric model. In *CVPR*, pages 1688–1695, 2010. 1
- [11] H. C. Longuet-Higgins and K. Prazdny. The interpretation of a moving retinal image. *P. Roy. Soc. B-Biol. Sci.*, 208:385–397, 1980. 3
- [12] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004. 2
- [13] T. Mörwald, J. Prankl, A. Richtsfeld, M. Zillich, and M. Vincze. BLORT - The Blocks World Robotic Vision Toolbox. In *ICRA*, 2010. 1, 6
- [14] F. Mosteller and J. Tukey. *Data analysis and regression: A second course in statistics*. Addison-Wesley Reading, Mass., 1977. 4
- [15] R. A. Newcombe, A. J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon. KinectFusion: real-time dense surface mapping and tracking. In *IEEE ISMAR*, pages 127–136, Oct. 2011. 1, 7
- [16] K. Pauwels, M. Tomasi, J. Díaz, E. Ros, and M. Van Hulle. A comparison of FPGA and GPU for real-time phase-based optical flow, stereo, and local image features. *IEEE T. Comput.*, 61(7):999–1012, 2012. 2
- [17] V. Prisacariu and I. Reid. PWP3D: Real-time segmentation and tracking of 3D objects. *IJCV*, 98:335–354, 2012. 1, 5, 6, 7
- [18] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In *CVPR*, pages 1297–1304, June 2011. 1
- [19] Wikipedia. Kinect, 2013. 7
- [20] C. Wu. SiftGPU: A GPU implementation of scale invariant feature transform (SIFT). <http://cs.unc.edu/~ccwu/siftgpu>, 2007. 2
- [21] C. Yang and G. Medioni. Object modelling by registration of multiple range images. *Image Vision Comput.*, 10(3):145–155, 1992. 3