2D Or Not 2D: Bridging the Gap Between Tracking and Structure from Motion

Karel Lebeda, Simon Hadfield, Richard Bowden
{K.Lebeda, S.Hadfield, R.Bowden}@surrey.ac.uk

University of Surrey, Guildford, GU2 7XH, United Kingdom

Abstract. In this paper, we address the problem of tracking an unknown object in 3D space. Online 2D tracking often fails for strong out-of-plane rotation which results in considerable changes in appearance beyond those that can be represented by online update strategies. However, by modelling and learning the 3D structure of the object explicitly, such effects are mitigated. To address this, a novel approach is presented, combining techniques from the fields of visual tracking, structure from motion (SfM) and simultaneous localisation and mapping (SLAM). This algorithm is referred to as TMAGIC (Tracking, Modelling And Gaussian-process Inference Combined). At every frame, point and line features are tracked in the image plane and are used, together with their 3D correspondences, to estimate the camera pose. These features are also used to model the 3D shape of the object as a Gaussian process. Tracking determines the trajectories of the object in both the image plane and 3D space, but the approach also provides the 3D object shape. The approach is validated on several video-sequences used in the tracking literature, comparing favourably to state-of-the-art trackers for simple scenes (error reduced by 22 %) with clear advantages in the case of strong out-of-plane rotation, where 2D approaches fail (error reduction of 58 %).

1 Introduction

Monocular, model-less, visual tracking aims to estimate the pose of an unknown object in every frame of a video-sequence (or to report its absence), given a bounding box containing the object in the first frame. This is a challenging task since the object appearance often changes significantly during the sequence. Many approaches attempt to learn changes in appearance, some of which come from object pose that cannot be modelled by a simple planar transformation. In this work, we directly address changes caused by a viewpoint variation. Instead of treating it as an "appearance change problem", we model explicitly the 3D information.

Many approaches have been proposed to overcome variations of appearance. Online approaches typically assume that the tracking has thus far succeeded, using this to enrich the representation of the object over time. The object is usually represented as a 2D patch [1, 2], a cloud of 2D points [3, 4] or a combination of these [5]. Unfortunately, variations of viewpoint lead to rapid changes in
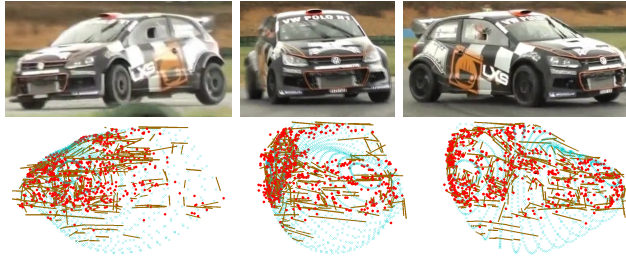
Fig. 1. Out-of-plane rotation change the object appearance significantly, here is a complete change in just 50 frames. First row: original images. Second row: feature cloud and final model returned by the tracker. Notice the bottom and back side of the car, which have not been observed yet, so the point cloud does not reach there and the model is smoothly extrapolated.

appearance – see Figure 1 for an example. This causes problems for 2D trackers which do not have sufficient observations to confidently update their object representation.

We take a different approach to this problem. Our proposed tracker explicitly models out-of-plane rotations, which proves beneficial, as they improve the numerical conditioning (wider baseline). The 3D shape of the object is estimated online using techniques developed in the fields of *Structure-from-Motion* (SfM) and *Simultaneous Localisation And Mapping* (SLAM). As such, this work can be seen as a bridge between visual tracking and SfM/SLAM, combining 2D feature tracking and object segmentation with camera pose and 3D point/line estimation, while avoiding the need for initialisation in SLAM [6]. Another difference from SfM/SLAM is object/background segmentation, where only a small portion of the image can be used.

In addition, we present a novel approach to modelling the object 3D shape using a *Gaussian Process* (GP). The model helps us distinguish which parts of the image belong to the projection of the object and which are background, allowing intelligent detection of new features (Section 3.1). In addition, the GP shape model 1) provides an initialisation of the 3D positions for newly detected 2D features, 2) mitigates the sparsity of features, 3) the surface normals of the GP indicate which points on the object may be visible to a particular camera and 4) the GP offers a model of the surface for visualisation and subsequent tasks, such as dense reconstruction or robot navigation, etc.

The rest of this work is structured as follows. In Section 2 related literature is reviewed. We then describe our tracker in Section 3. It is experimentally evaluated in Section 4, while Section 5 draws conclusions.

2   Related Work

There is a large body of literature in 2D tracking, SfM and SLAM. Here we summarise the most relevant work in each field, as well as recent *state of the art* approaches.

Visual Tracking: One of the most influential works in the field of visual tracking is undoubtedly the Lucas-Kanade tracker (LK) [7]. This technique iteratively matches image patches by linearising the local image gradients and minimising the sum of squared pixel errors. Many current trackers follow and extend this idea by adding an upper layer managing a cloud of LK *tracklets* [4, 8]. One recent example is the Local-Global Tracker (LGT) [3] by Cehovin *et al.*, using a coupled-layer visual model. The local layer consists of a set of visual patches, while the global layer maintains a probabilistic model of object features such as colour, shape or motion. The global model is learned from the local patches and in turn it constrains the addition of new patches. FLOTrack (Featureless Objects Tracker) [9] uses a similar two layer approach, however edge features are used instead of conventional points to increase robustness to lack of texture.

Another approach to 2D tracking is *tracking by detection* where tracking is defined as a classification task. Grabner *et al.* [10] employed an *online boosting* method to update the appearance model while minimising error accumulation (drift). Babenko *et al.* [11] use *multiple instance learning*, instead of traditional supervised learning, for a more robust tracker. ℓ_1-norm tracker [12] learns a dictionary from local patches to handle occlusions. Kalal *et al.* combined tracking with detection in their Tracking-Learning-Detection (TLD) [5] framework, where (in)consistency of tracker and detector helps to indicate tracking failure. There are also many successful approaches using *particle filtering* for tracking. One of the most notable is CONDENSATION (Conditional Density Propagation) [13], which brought into the field of visual tracking the use of particles for non-parametric modelling of a pose probability distribution. Ross *et al.* [2] extended particle filtering by introducing *incremental learning* of an object appearance subspace that allowed the model to adapt to changes.

3D monocular tracking typically employs 3D models of the object. These approaches work with a user-supplied or previously learned model of a known object, which is then tracked in the sequence, examples of which are the tracking of the pose of human bodies, or vehicles in traffic scenes [14–17] or general *boxes* [18]. Another body of literature deals with 3D reconstruction of deformable surfaces [19]. However, we will not address these areas further as our focus is on model-less tracking. There have been relatively few attempts at learning 3D tracking models on the fly, but such approaches are fundamental to SfM. An example of a recent model-less 3D tracker is the work of Feng *et al.* [20], who employ SLAM techniques similar to TMAGIC, together with object segmentation. However, they do not attempt to estimate the surface shape beyond just a cloud of points, which allows the more advanced visibility reasoning used in TMAGIC. Another similar approach is the work of Prisacariu*et al.* [21] who uses level-set techniques for object segmentation. Our approach uses both point and line features to increase robustness to lack of texture.

Structure from Motion: *Structure from motion* (SfM) is an area of great interest for the computer vision community. Here, the task is to simultaneously estimate the structure of a scene observed by multiple cameras, and the poses of these cameras. Pictures of the scene can be either ordered in a sequence [22–24]

or unordered [25, 26]. Great advances have been recently achieved in the size and automation of the SfM process by Snavely *et al.* in their Bundler system [26, 27] (based on Bundle Adjustment, BA). Their approach is *sequential*, i.e. starting reconstruction with a smaller number of cameras and then adding progressively more. A different approach has been taken by Gherardi *et al.* [25], who merge smaller reconstructions hierarchically. Pan *et al.* [22] reached real-time speed of SfM in their ProFORMA system for guided scanning of an object. This work is also very related to ours, however the object/background separation allows us to avoid the assumption that the object occupies the majority of the image.

A sparse reconstruction can be upgraded to dense as a post-processing step, or alternatively a dense reconstruction can be performed directly [28]. Recently, the approach of Garg *et al.* [29] used variational optimisation from dense point-trajectories to directly estimate dense non-rigid SfM.

Visual SLAM: *Simultaneous localisation and mapping* is an area commonly associated with robotics, with similarities to SfM. The main difference is that SLAM, originally used for navigation of robots, is an online process. Therefore the images are always ordered in a sequence. It further differs in that the reconstructed cloud of features is usually sparse and the reconstructed area larger.

The first monocular vision-based SLAM technique was MonoSLAM by Davison *et al.* [30]. It was later extended to work with straight lines [31, 32]. These algorithms (based on the Extended Kalman Filter, EKF) were able to run in real time and typically managed a very low number of features in the point cloud. Recently, there has been a shift from EKF to BA [33, 34] and from sparse to dense features [35], ultimately leading to whole-image-alignment works [36, 37].

3   Algorithm Description

We seek to track a 3D object throughout a sequence learning a model of appearance on the fly. However, our model differs from most online tracking approaches in that we employ ideas from both SfM and SLAM to form a 3D representation of the model that can cope with out-of-plane rotation. The program and data flow of the TMAGIC tracker is as follows (illustrated in Figure 2, see the subsequent sections for full descriptions of the individual elements). The tracking loop is performed on every frame. 2D features (points and line segments in the image) are tracked in the new frame (Section 3.1), yielding the sets of features currently visible. Using these, the new camera pose can then be estimated (Section 3.2), while keeping the corresponding 3D features (points and lines in the real world) fixed. The world coordinate system is not fixed, so we can safely assume that the camera is moving around a stationary object. The tracking loop is repeated until a change in viewpoint necessitates an update of the 3D features, using the modelling subsystem.

The first step of modelling is a Bundle Adjustment (BA, Section 3.3). This refines the positions of 3D features and the camera, using the 2D observations. The updated features are subsequently used to retrain the shape model (Section 3.4), which can be exploited in two ways. The model defines regions of
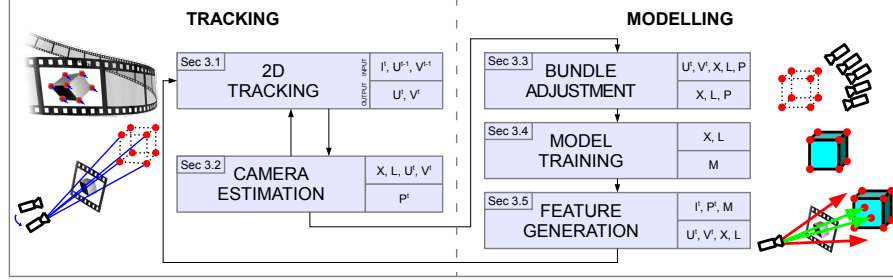
Fig. 2. Overview of the TMAGIC tracker. Symbols on the right side of each element indicate inputs and outputs, as labelled for 2D tracking. While the tracking loop is repeated in every frame, modelling only runs when necessary (according to Eq. (3)).

the image which are eligible to detect new 2D features. Secondly, it provides an initialisation of the corresponding backprojected 3D features (Section 3.5). Features, successfully extracted using the current frame, camera pose and the shape model, enrich the 2D and 3D sets for use in future tracking.

3.1   2D Features and Tracking

The TMAGIC algorithm uses two types of features: points and line segments. The main advantages of point features are that they form readily available unique descriptors (patches), are localised precisely and have intuitive and simple projective properties. On the other hand, line features, which provide complementary information about the image, have different virtues. Lines encode a higher level of structural information [38], e.g. constraining the orientation of the surface. They can be not only texture-based, but also stemming from the shape of the object [9]. Therefore in man-made environments they appear in situations where point features are scarce [31].

The 2D point features $\mathbf{u}_i^t \in \mathcal{U}^t$ are extracted using two techniques: Difference of Gaussians and Hessian Laplace [39]. These features are tracked independently by an LK tracker from frame $\mathtt{I}^{t-1}$ to $\mathtt{I}^t$ ($\mathtt{I}^t$ represents the current frame and derived measurements, such as an intensity gradient). Features, which do not converge are removed from the 2D feature cloud. The tracked features generate a set of correspondences $\{(\mathbf{u}_i^{t-1}, \mathbf{u}_i^t)\}$, which are subsequently verified by LO-RANSAC [40, 41]. Outliers to RANSAC, i.e. correspondences inconsistent with a global epipolar geometry model, are removed, as well as their respective 3D features, as these are likely to lie on the background.

The 2D line features $\mathbf{v}_i^t \in \mathcal{V}^t$ are extracted using the LSD [42] approach, a line segment detector with false-positive detection control. Lines are tracked as follows. Firstly, the LSD is executed on $\mathtt{I}^t$ to obtain a set of candidate segments $\mathbf{v}_j'^t \in \mathcal{V}'^t$. Along each of the previous line segments $\mathbf{v}_i^{t-1}$ a number of edge points are then sampled. Each of these is tracked independently, using the *guided edge search* of [9], leading to a new edge point in the current frame. If this new point belongs to a line segment in $\mathcal{V}'^t$, it votes for it. The segment $\mathbf{v}_j'^t$ with

the most votes becomes the new feature $\mathbf{v}_i^t$. As there is no way to validate line correspondences w.r.t. an epipolar geometry [43], the features are validated using a threshold on the minimum number of votes (3 out of 5).

3.2   Camera Estimation

The camera with a projection matrix $\mathtt{P}^t$ is defined by the rotation $\mathtt{R}^t$ and position $\mathbf{C}^t$ of the projection centre in the world coordinate frame, given by the decomposition $\mathtt{P} = \mathtt{KR}\left[\mathtt{I}_{3\times3}| - \mathbf{C}\right]$, where $\mathtt{K}$ is a calibration matrix of intrinsic camera parameters [43]. For simplicity, we define a general projection function $\mathtt{F}_\mathtt{P}$, such that 3D lines are projected as $\mathbf{v}_i^t = \mathtt{F}_{\mathtt{P}^t}(\mathbf{L}_i)$ and 3D points as $\mathbf{u}_i^t = \mathtt{F}_{\mathtt{P}^t}(\mathbf{X}_i)$.

Assuming we have a cloud of 3D features (points $\mathbf{X}_i \in \mathcal{X}$ and line segments $\mathbf{L}_i \in \mathcal{L}$, which are defined by their end-points, specified in Section 3.3) and their projections $(\mathcal{U}^t, \mathcal{V}^t)$, it is possible to estimate a pose $(\mathtt{R}, \mathbf{C})$ of the camera $\mathtt{P}^t$. We do not attempt to compute the calibration matrix $\mathtt{K}$ of the camera exactly, instead we use an estimate based on image dimensions [23]:

$$\mathtt{K} = \begin{bmatrix} w + h & 0 & w/2 \\ & w + h & h/2 \\ & & 1 \end{bmatrix} \tag{1}$$

where w and h are the width and height of the image respectively. This formula would not suffice for cases of strong zoom, wide-angle cameras, or cropped videos (shift of the principal point and narrowed viewing angle). However, Pollefeys *et al.* [23] show that images obtained by standard cameras are generally well approximated by this formula.

Estimation of calibrated camera pose given 2D to 3D point correspondences is a standard textbook problem, e.g. solved by a P3P (Perspective-3-Points [43]) RANSAC. However, there is no simple generalisation of the P3P problem for lines. Therefore we use an optimisation approach to solve for camera pose:

$$\mathtt{P}^t = \arg\min_{\bar{\mathtt{P}}^t} \left(\sum_{i=1}^{|\mathcal{U}^t|} ||\mathbf{u}_i^t - \mathtt{F}_{\bar{\mathtt{P}}^t}(\mathbf{X}_i)||^2 + \sum_{i=1}^{|\mathcal{V}^t|} \left((\widetilde{\boldsymbol{\mu}}_{\mathbf{v}_i^t}^\top \widetilde{\mathbf{l}}_i^t)^2 + (\widetilde{\boldsymbol{\nu}}_{\mathbf{v}_i^t}^\top \widetilde{\mathbf{l}}_i^t)^2 \right) \right) , \tag{2}$$

using both point and line features in a unified framework and exploiting the sequential nature of tracking. The error function consists of an error term for each point and line feature (in the first and second summation, respectively). For points, this is just a norm of the projection error. For line features, we define the error terms as orthogonal distances of the end-points of the segment $\mathbf{v}_i^t$ ($\widetilde{\boldsymbol{\mu}}_{\mathbf{v}_i^t}, \widetilde{\boldsymbol{\nu}}_{\mathbf{v}_i^t}$, in homogeneous coordinates), to the projection of the 3D line $\widetilde{\mathbf{l}}_i^t = \mathtt{F}_{\bar{\mathtt{P}}^t}(\mathbf{L}_i)$ (homogeneous, normalised to the unit length of the normal vector). Note that since the line may not be fully visible (and is theoretically infinite), we use only perpendicular distances to cope with the *aperture problem* [9, 31]. This minimisation is initialised at the pose in the previous frame ($\mathtt{P}^{t-1}$). During our experiments, it was found that due to the smooth nature of the derivatives of (2), the basin of convergence for this optimisation is several orders of magnitude larger than typical inter-frame difference. See supplementary material for derivation of the error function Jacobian used.
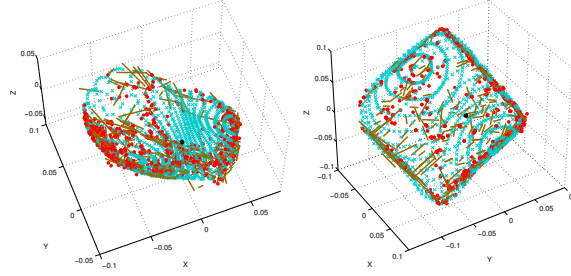
Fig. 3. Examples of 3D feature clouds and GP-learned smooth models. See the supplementary material for more visualisations. Notice the unseen parts of the objects, which are without features and where the model is extrapolated (i.e. the rear side of the car and the "pole" on the left side of the cube).

In the first frame, we choose the world coordinate frame (which is defined only up to a similarity) as follows. The object, initialised as a sphere (see Section 3.4 for more details), is centred at the origin and camera centre $\mathbf{C}^1$ is at $(0,0,1)^\top$. The rotation $\mathtt{R}^1$ is set such that the origin is projected to the centre of the user-given bounding box and the y axis of the camera coordinate system is parallel to the y-z plane of the world coordinate system (see Figure 4).

3.3   Bundle Adjustment

After initialisation, 2D tracking is performed until the distance between camera centres exceeds a specified threshold $\theta_\mathbf{C}$ [22]:

$$||\mathbf{C}^t - \mathbf{C}^{t'}|| > \theta_\mathbf{C} \ , \tag{3}$$

where t' is the time of the last BA. When this condition is satisfied, the modelling part of the algorithm is performed. Firstly, the bundle adjustment refines the positions of 3D features $\mathcal{X}, \mathcal{L}$ and cameras $\mathcal{P}$ (for the purposes of BA, we define $\mathcal{P}$ as the set of previous cameras $\mathtt{P}^1, \ldots, \mathtt{P}^t$). If speed is an issue, one may limit the BA to take only the last k cameras into account, i.e. to use $\mathcal{P}' = \{\mathtt{P}^i | i \in [\max(1, t-k); t]\}$, however, in our experiments this did not prove necessary (thus we set $k = \infty$). The BA [44] minimises a similar error to (2):

$$\underset{\mathcal{X}, \mathcal{L}, \mathcal{P}}{\arg\min} \sum_{u=1}^{t} \left(\sum_{i=1}^{|\mathcal{U}^u|} ||\mathbf{u}_i^t - \mathrm{F}_{\bar{\mathtt{P}}^t}(\mathbf{X}_i)||^2 + \sum_{i=1}^{|\mathcal{V}^u|} \left((\widetilde{\boldsymbol{\mu}}_{\mathbf{v}_i^t}^\top \widetilde{\mathbf{l}}_i^t)^2 + (\widetilde{\boldsymbol{\nu}}_{\mathbf{v}_i^t}^\top \widetilde{\mathbf{l}}_i^t)^2 + \rho_i^2 \right) \right) \ , \tag{4}$$

where the added term ρ_i is a regularisation term, which ensures that the lengths of 3D line segments are close to those observed. Note also that every point from $\mathcal{U}^t$ and $\mathcal{V}^t$ has a correspondence in $\mathcal{X}$ and $\mathcal{L}$ for every t, but not necessarily vice-versa, due to points which are not currently visible.

3.4   Gaussian Process Modelling

As discussed previously, the object shape is modelled as a Gaussian Process [45, 46]. This allows us to infer a fully dense 3D model from the finite collection

of discrete observations $\mathcal{X}$ and $\mathcal{L}$. Using the GP in this manner can be seen as estimating a distribution over an infinite number of possible shapes. The expectation of such a distribution (the most probable shape) can be used to model the object, while the variance at any point represents confidence.

The obtained model is non parametric, i.e. it can fit to any type of object without needing reparameterisation. The probabilistic nature of the model prevents overfitting through an implicit "Occam's-razor" effect, that favours models which are both simple and which explain the observations well.

In TMAGIC, we use a parameterisation of the shape as follows. The observed 3D points (point features $\mathcal{X}$ and end-points[1] of line features $\mathcal{L}$) are first re-expressed as a vector from the centre of mass of the object ($\mathbf{Y}_i \in \mathcal{Y}$). We then use these as training points, where the unit-length normalised vectors $\hat{\mathbf{Y}}_i = \mathbf{Y}_i/||\mathbf{Y}_i||$ represent the independent variable and the radii $r_{\hat{\mathbf{Y}}_i} = ||\mathbf{Y}_i||$ the dependent variable. As it does not suffer from a singularity in any direction, we found this parameterisation superior to alternatives such as spherical coordinates (predicting a radius from azimuth and elevation), despite the higher dimensionality. This representation prevents us from modelling complicated shapes (e.g. extreme concavities), however it proves sufficient in our experiments.

Without loss of generality, we can assume that the centre of mass of the training points coincides with the origin of the world coordinate system. In this case, for a query direction $\hat{\mathbf{Q}}$ (where $||\hat{\mathbf{Q}}|| = 1$), the resulting 3D point $\mathbf{Q}$ is predicted as [45]:

$$
\begin{aligned}
\mathbf{Q} = \hat{\mathbf{Q}} \Big[&\mathrm{K}(\hat{\mathcal{Y}}, \hat{\mathbf{Q}})^\top \mathrm{K}(\hat{\mathcal{Y}}, \hat{\mathcal{Y}})^{-1} \mathbf{r}_{\hat{y}} \\
&\pm \mathrm{K}(\hat{\mathcal{Y}}, \hat{\mathbf{Q}})^\top \mathrm{K}(\hat{\mathcal{Y}}, \hat{\mathcal{Y}})^{-1} \mathrm{K}(\hat{\mathcal{Y}}, \hat{\mathbf{Q}}) \Big] ,
\end{aligned} \tag{5}
$$

or more succinctly as $\mathbf{Q} = \hat{\mathbf{Q}} \left(r_{\hat{\mathbf{Q}}} \pm \sigma_{\hat{\mathbf{Q}}} \right)$, where $r_{\hat{\mathbf{Q}}}$ is the predicted radius and $\sigma_{\hat{\mathbf{Q}}}$ is the confidence. K is the kernel function of the GP, relating to the surface properties of the modelled object, and may be any positive definite two-parameter function. This function is learned during tracking, such that the likelihood of the training data is maximised. The notation $\mathbf{r}_{\hat{y}}$ represents a vector of norms of all vectors in the training set $\mathcal{Y}$, i.e. $\mathbf{r}_{\hat{y};i} = r_{\hat{\mathbf{Y}}_i} = ||\mathbf{Y}_i||$.

Intuitively, Equation (5) shows that the predicted radius at any point is defined by the training radii while accounting for the spatial relationships between the data points. The variance relates to these spatial relationships. The influence of any particular element of the training data is quantified by $\mathrm{K}(\hat{\mathcal{Y}}, \hat{\mathbf{Q}})$, while $\mathrm{K}(\hat{\mathcal{Y}}, \hat{\mathcal{Y}})^{-1}$ removes any correlation within the training data.

We tried several alternative approaches for the surface shape modelling, such as mesh-based or modelling as a parametric probability distribution. However, none had the properties required. The benefits of the GP include the ability to model a wide range of shapes without prior knowledge (non-parametric), no overfitting, good theoretical foundation and its probabilistic nature, giving confidence estimates across the object.

---

[1] It is possible to sample more points along line features which have high confidence.
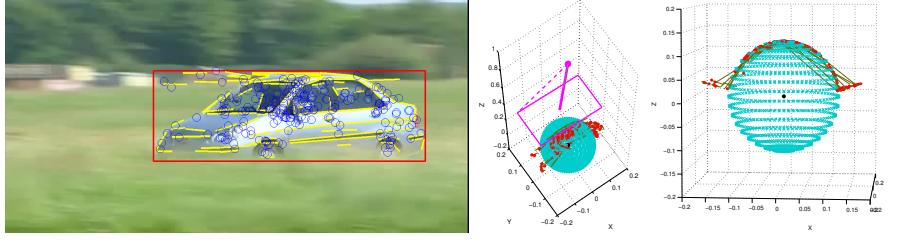
Fig. 4. State of the proposed TMAGIC tracker after the first frame. Left: a bounding box, $\mathcal{U}^1$ and $\mathcal{V}^1$. Right: $\mathcal{X}, \mathcal{L}$, initial model M (the blue sphere) and $\mathtt{P}^1$ (magenta). For the camera, the visualisation shows the projection centre $\mathbf{C}$, principal direction and image plane. The upper left corner of the image plane is indicated by the dashed line.

3.5 Feature generation

For camera pose estimation (Sec. 3.2), we assumed that the 3D feature clouds $\mathcal{X}$ and $\mathcal{L}$ are known. In this section, we address the issue of feature generation and localisation. The assumption is made that a shape model of the object is given.

In the first frame $\mathtt{I}^1$, initial sets of 2D features $\mathcal{U}^1$ and $\mathcal{V}^1$ are generated inside a user-supplied bounding box. When generating a 3D feature $\mathbf{X}_i$ for a new 2D point $\mathbf{u}_i^t$ (in the case of line features, both end-points must lie on the surface), the process is as follows. Firstly, the corresponding ray $\mathbf{Z}$ (parameterised by $\bar{\alpha}$) from the camera centre is generated:

$$\mathbf{Z}(\bar{\alpha}) = \mathbf{C} + \bar{\alpha}(\mathtt{KR})^{-1}\widetilde{\mathbf{u}}_i^t \qquad (\bar{\alpha} > 0) , \tag{6}$$

where $\widetilde{\mathbf{u}}_i^t$ is a homogeneous representation of $\mathbf{u}_i^t$. Then a search for an intersection between the ray and the shape is performed:

$$\mathbf{X}_i = \mathbf{Z}(\alpha) , \tag{7}$$

$$\alpha = \underset{\bar{\alpha}>0}{\arg\min} \, \bar{\alpha} \qquad \text{s.t.} ||\mathbf{Z}(\bar{\alpha})|| = r_{\hat{\mathbf{Z}}(\bar{\alpha})} . \tag{8}$$

If the minimisation of (8) has no solution, it means that the ray does not intersect the *mean surface* given by the GP (from the distribution of possible surfaces). The use of the mean surface corresponds to a threshold such that there is an equal probability of false positives (a detection on the background was added to the feature set) and false negatives (a point on the object surface was rejected). If there is prior knowledge about the respective robustness of other components available, this can be exploited by adding the appropriate factor (multiple of $\sigma_{\hat{\mathbf{Z}}(\bar{\alpha})}$) to $r_{\hat{\mathbf{Z}}(\bar{\alpha})}$ in (8).

Thus far, the process has been the same both for initialisation in the first frame and for adding new features after model retraining in the subsequent frames. However, there are several differences. Firstly, if a ray does not intersect the surface during the generation of new features in frame $f > 1$, it is not used (features are detected over the whole image). However, in the first frame, all the 2D features will lie inside the user-specified bounding box. In this case, we

reconstruct them such that they minimise the distance to the surface (even when they do not intersect), leading to the fringe seen in Figure 4:

$$\alpha = \underset{\bar{\alpha}>0}{\arg\min}(||\mathbf{Z}(\bar{\alpha})|| - r_{\hat{\mathbf{Z}}(\bar{\alpha})})^2 \ . \tag{9}$$

Generation of new features in $f > 1$ has one further condition. Since adding new features increases the time complexity of all other computations, new features are added only into uncertain regions of the object (with variance greater than a specified threshold, $\sigma_{\hat{\mathbf{Z}}(\alpha)} > \theta_\sigma$).

As previously mentioned, some of the 3D features may be temporarily occluded, i.e. without 2D correspondences. Surface normals, given by the shape model, provide us with one tool to determine which parts of the object are (not) visible from a particular direction. TMAGIC uses this information in two complementary ways. Firstly, if a 3D feature is deemed not visible, but it has 2D correspondence (e.g. adheres to an object contour), it can be removed. On the other hand, if a 3D feature has no 2D correspondence, but is on a surface which is seen by $\mathtt{P}^t$ under an angle close to normal, the 2D feature can be redetected. This is performed by projecting it into $\mathtt{I}^t$ by $\mathtt{P}^t$ and then tracking it in 2D using a stored appearance patch. *Loop closures* (as termed in the SLAM literature) are thus possible when a number of previously seen features are redetected.

4   Experimental Evaluation

In all experiments, the parameters were fixed as follows: $\theta_{\mathbf{C}} = 10\,\%$, $\theta_\sigma = 0.5\,\%$, relative to the scene size. Our proof-of-concept implementation currently runs in seconds per frame. However, there are possibilities for trivial technical improvements and for parallelisation, allowing real-time application.

Synthetic data: Firstly, we show results on a synthetic sequence CUBE1 (Fig. 5). This has been rendered to have the following properties. It contains a cube, rotating with speed 1°/frame. Some of the sides are rich in texture, some are weakly textured. From the point of view of our tracker, the camera circles around the fixed cube with a perfect circular trajectory (see Fig. 6). However, since the

Fig. 5. Selected frames from the CUBE1 sequence. Notice, how TMAGIC learns the new face of the cube. #194: unknown shape, the surface is smoothed over. #213: first features detected, shape roughly estimated. #239: more features identified, shape refined. #272: Final state, model in agreement with the object.
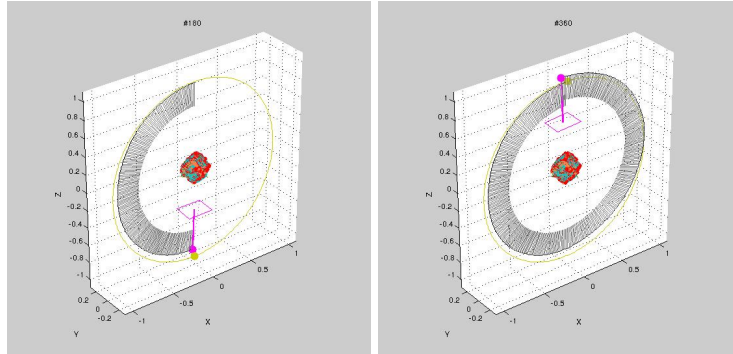
Fig. 6. The 3D scene in $t = 180$ and 360. The camera and features are shown in the same way as in Figure 4, the camera trajectory (centres and principal directions for each previous frame) is shown in black. Ground-truth trajectory is in yellow. The details of the model can be seen in Figure 3.
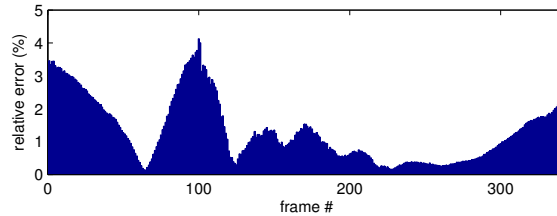

Fig. 7. Deviation of the trajectory from a perfect circle (least-squares fitted). Errors are relative to the radius of the circle. The trajectories at the beginning and end of the sequence did not meet (due to accumulated error prior to loop closure), thus the deviation from the fitted ground truth is distributed between the two. The peak around frame #100 is due to a temporary inaccuracy during the transition between sides of the cube, when the continuously visible side is lacking visual features. However, TMAGIC recovers once sufficient visual evidence has been accumulated.

world coordinate frame is defined only up to a similarity transform and can be moved freely during the BA, it is not possible to measure quality of a tracker directly w.r.t. this expected position (i.e. no absolute ground truth is possible). Therefore we fit a 3D circle to the points and measure the error as an orthogonal distance from this circle. Fig. 7 shows and explains the results. If we assume the camera orbits at a distance of 1 m, the mean camera pose error is 1.3 cm. This indicates a very close approximation to the circular trajectory. Despite being based on sparse data, the learned shape model represents the cubic shape (of side equal to approximately 17 cm) accurately, having mean reconstruction error 3.4 mm. See supplementary material for the input video and results.

To compare our method with currently used reconstruction approaches, we processed this sequence (with no background to account for) with VisualSfM [47, 48] and Bundler [27, 49]. While Bundler surprisingly failed, reconstructing 2 separate cubes, VisualSFM performs worse than TMAGIC with a comparable reconstruction error of 2.8 mm, but 72 % larger camera trajectory error of 2.3 cm.
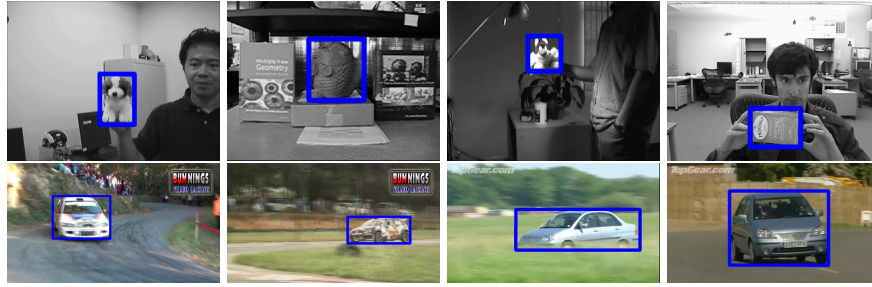
Fig. 8. First frames from the test sequences. From top and left: Dog, Fish, Sylvester, Twinings, Rally-Lancer, Rally-VW, TopGear1 and TopGear2. The initial bounding boxes are overlaid.

Real data: The performance of TMAGIC was further analysed on several sequences, used in previous 2D visual tracking publications. These sequences contain visible out-of-plane rotation in all cases. Additionally, we use several new sequences of drifting cars, which have, besides strong motion blur, significant out-of-plane rotation as a challenge. The first frames are shown in Figure 8.

We compared our tracker with several state of the art tracking algorithms in Table 1: LGT [3], TLD [5] and FoT [8]. The FoT (Flock of Trackers) tracker is similar to our approach, in that it employs a group of independently tracked features with a higher management layer, however in 2D only. The next columns show the effect of the different stages of TMAGIC on performance. 3D tracking (**T**) assumes a fixed 3D model (sphere) and feature locations, added reconstruction (**TMIC**) allows updating of the feature clouds and a primitive object inference (a rigid sphere fit). Finally, TMAGIC gives performance for the full system. The performance metric was *localisation error*, i.e. the distance of the centre of the bounding box to the ground-truth centre. The results are visualised in Figure 9 and the mean values tabulated in Table 1. Additionally, we show mean overlap of the tracked and ground-truth bounding boxes.

Table 1. Tracking results: mean localisation error/mean overlap. Bold numbers indicate the best result, underlined numbers the second best. See the text for discussion.

|	LGT	TLD	FoT	T	T-MIC	TMAGIC
Dog [50]	19.3/20	12.3/ 56	**4.7**/ 63	13.9/65	9.3/**71**	12.1/46
Fish [2]	15.6/20	8.8/ 72	9.2/**75**	8.0/68	10.4/ 65	**5.6**/69
Sylvester [2]	**13.1**/16	18.0/**58**	18.0/**58**	37.3/42	35.3/ 9	17.8/48
Twinings [11]	22.5/18	13.2/ 38	15.5/ 44	42.9/31	16.2/ 29	**9.1**/**53**
Rally-Lancer	145.8/19	333.5/ 13	734.5/ 13	127.8/43	121.3/ 42	**94.3**/**53**
Rally-VW	91.3/21	152.5/ 48	196.9/ 39	149.4/39	148.3/ 39	**47.6**/**62**
TopGear1	**16.3**/49	65.1/ 34	80.4/ 41	39.0/49	44.5/ 43	40.0/**56**
TopGear2	84.3/37	104.3/ 21	117.9/ 29	48.4/51	84.1/ 31	**34.7**/**59**
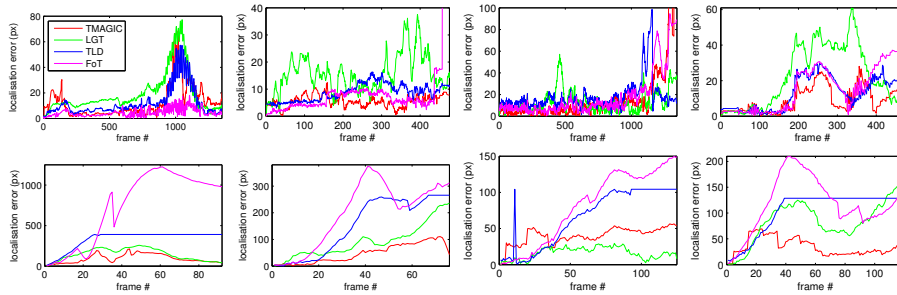
Fig. 9. Visualisation of results of the quantitative performance analysis. From top and left: Dog, Fish, Sylvester, Twinings, Rally-Lancer, Rally-VW, TopGear1 and TopGear2.

On the Dog sequence, the TMAGIC and TLD trackers perform similarly, and LGT slightly worse. All these trackers experience difficulties at about frame #1000, where the dog is partially occluded by the image border. This is however not a problem for FoT, which estimates the position accurately even under such strong occlusion. The Fish sequence is relatively easy, with all the trackers reaching low errors and TMAGIC being the best one. On the Sylvester scene, TMAGIC as well as FoT track consistently well until the end. Both LGT and TLD have similar momentary failures (frames #450 and #1100, respectively) but both are able to recover. For LGT, the duration of the problematic part of the sequence is shorter and the error is smaller, rendering it the best tracker for this sequence. The Twinings sequence contains full rotation and was originally created to measure trackers' robustness to out-of-plane rotation [11]. Unsurprisingly, TMAGIC significantly outperforms the state of the art on this sequence. The average localisation error reduction for all these scenes is 22 %.

The cars sequences are chosen because they contain rigid 3D objects under strong out-of-plane rotation (around 180°) with significant camera motion. Therefore the TLD and FoT trackers, which are trying to track a plane only (one side of the car) instead of the 3D object, fail. As the cars rotate, the tracked parts are no longer usable and TLD reports this (the horizontal sections in Figure 9). FoT is incapable of reporting object disappearance and it attempts to continue tracking, exacerbating the situation. The LGT tracker, which has a less rigid model of the object, is sometimes capable of tracking after the cars start to rotate, if the rotation is slow enough for the 2D shape model to adapt. The TMAGIC tracker is also able to adapt as the object rotates, and explicitly modelling the car in 3D improves robustness by allowing us to intelligently detect new features. While 2D trackers attempt to mitigate the effects of out-of-plane rotation, TMAGIC actively exploits it. This gives it a significant edge, resulting in the localisation error being reduced by 58 % on average. Notice that the errors in the Rally sequences are generally higher, due to the higher resolution. The resulting model for the Rally-VW sequence is visualised in Figures 1 and 3. The car is modelled accurately, except for missing elements at the rear of the vehicle, which have not been observed during the sequence.

The last three columns of Tab. 1 show the effect of the different stages of TMAGIC on performance. Firstly, we compare FoT with T (2D and 3D trackers based on the same principle). FoT performs better on sequences without rotation (higher accuracy of the solution) and the advantage of 3D tracking becomes apparent with stronger out-of-plane rotations (decreasing the error up to five-fold). The next step is performing refinement of the 3D features and fitting a naïve spherical model (T→TMIC). However, the effect of this procedure on the performance of tracking in the image plane is imperceptible, despite the improved plausibility of the feature cloud. The final stage is training a more complex shape model using the feature cloud (TMIC→TMAGIC). This yields the most significant improvement (average error reduction of 50 %), rendering TMAGIC by far the best of the evaluated trackers in cases of out-of-plane rotations. In the case of TOPGEAR1, mostly the front part of the car is being modelled, shifting the centre of the bounding box forward and therefore adversely affecting the final results.[2]

5   Conclusion

The experiments show that the TMAGIC (Tracking, Modelling And Gaussian-Process Inference Combined) tracker is able to track standard sequences, used in many previous publications, with a comparable performance to the state of the art. However, by explicitly modelling the 3D object, it handles out-of-plane rotations significantly better and can also track in cases of full rotation. TMAGIC consistently outperforms simpler variants (TMIC etc.), especially in scenarios when the object/background segmentation is vital. This shows the benefit of the shape model, used for filtering features and initialisation of their 3D positions.

TMAGIC works under the assumption that the object is rigid. TMAGIC is robust to small shape variations (e.g. a face), but is not capable of tracking articulated objects, e.g. a walking person, and extension to non-rigid object tracking is the subject of future work. One possible approach would be storing multiple models to span the area of possible shapes. This can also help with redetection to recover from drift. Another limitation would be full occlusions in long-term tracking. An additional redetection stage (as in TLD and other long-term trackers) could fix this, but is beyond the scope of this paper. Fast motion may also cause failure in the underlying 2D tracking. However, the algorithm is robust to low textured objects through the use of line features (taken from texture-less tracking literature).

TMAGIC tracks a single object. A naive multi-object extension, running it multiple times in parallel, would be trivial since TMAGIC doesn't require any pre-learning and the object properties are estimated online. Advanced correlation or occlusion reasoning would be an interesting field for future research.

---

[2] See `cvssp.org/Personal/KarelLebeda/TMAGIC` for the sequences and more results.

References

1. Hare, S., Saffari, A., Torr, P.H.S.: Struck: Structured output tracking with kernels. In: ICCV. (2011)
2. Ross, D., Lim, J., Lin, R.S., Yang, M.H.: Incremental learning for robust visual tracking. IJCV (2008)
3. Cehovin, L., Kristan, M., Leonardis, A.: Robust visual tracking using an adaptive coupled-layer visual model. PAMI (2013)
4. Kolsch, M., Turk, M.: Fast 2D hand tracking with flocks of features and multi-cue integration. In: CVPRW. (2004)
5. Kalal, Z., Mikolajczyk, K., Matas, J.: Tracking-Learning-Detection. PAMI (2012)
6. Mulloni, A., Ramachandran, M., Reitmayr, G., Wagner, D., Grasset, R., Diaz, S.: User friendly SLAM initialization. In: ISMAR. (2013)
7. Lucas, B., Kanade, T.: An iterative image registration technique with an application to stereo vision. In: IJCAI. (1981)
8. Matas, J., Vojir, T.: Robustifying the flock of trackers. In: CVWW. (2011)
9. Lebeda, K., Matas, J., Bowden, R.: Tracking the untrackable: How to track when your object is featureless. In: ACCV DTCE. (2012)
10. Grabner, H., Leistner, C., Bischof, H.: Semi-supervised on-line boosting for robust tracking. In: ECCV. (2008)
11. Babenko, B., Yang, M.H., Belongie, S.: Robust object tracking with online multiple instance learning. PAMI (2011)
12. Xing, J., Gao, J., Li, B., Hu, W., Yan, S.: Robust object tracking with online multi-lifespan dictionary learning. In: ICCV. (2013)
13. Isard, M., Blake, A.: CONDENSATION - conditional density propagation for visual tracking. IJCV (1998)
14. Prisacariu, V., Segal, A., Reid, I.: Simultaneous monocular 2D segmentation, 3D pose recovery and 3D reconstruction. In: ACCV. (2012)
15. Dame, A., Prisacariu, V., Ren, C., Reid, I.: Dense reconstruction using 3D object shape priors. In: CVPR. (2013)
16. Sigal, L., Isard, M., Haussecker, H., Black, M.: Loose-limbed people: Estimating 3D human pose and motion using non-parametric belief propagation. IJCV (2012)
17. Wojek, C., Walk, S., Roth, S., Schindler, K., Schiele, B.: Monocular visual scene understanding: Understanding multi-object traffic scenes. PAMI (2013)
18. Kim, K., Lepetit, V., Woo, W.: Keyframe-based modeling and tracking of multiple 3D objects. In: ISMAR. (2010)
19. Varol, A., Shaji, A., Salzmann, M., Fua, P.: Monocular 3D reconstruction of locally textured surfaces. PAMI (2012)
20. Feng, Y., Wu, Y., Fan, L.: On-line object reconstruction and tracking for 3D interaction. In: ICME. (2012)
21. Prisacariu, V.A., Kahler, O., Murray, D.W., Reid, I.D.: Simultaneous 3D tracking and reconstruction on a mobile phone. In: ISMAR. (2013)
22. Pan, Q., Reitmayr, G., Drummond, T.: ProFORMA: Probabilistic feature-based on-line rapid model acquisition. In: BMVC. (2009)
23. Pollefeys, M., Van Gool, L., Vergauwen, M., Verbiest, F., Cornelis, K., Tops, J., Koch, R.: Visual modeling with a hand-held camera. IJCV (2004)
24. Tanskanen, P., Kolev, K., Meier, L., Camposeco, F., Saurer, O., Pollefeys, M.: Live metric 3D reconstruction on mobile phones. In: ICCV. (2013)
25. Gherardi, R., Farenzena, M., Fusiello, A.: Improving the efficiency of hierarchical structure-and-motion. In: CVPR. (2010)

26. Agarwal, S., Snavely, N., Simon, I., Seitz, S.M., Szeliski, R.: Building Rome in a day. In: ICCV. (2009)
27. Snavely, N., Seitz, S.M., Szeliski, R.: Modeling the world from internet photo collections. IJCV (2007)
28. Furukawa, Y., Ponce, J.: Accurate, dense, and robust multi-view stereopsis. PAMI (2010)
29. Garg, R., Roussos, A., Agapito, L.: Dense variational reconstruction of non-rigid surfaces from monocular video. In: CVPR. (2013)
30. Davison, A.J., Reid, I., Molton, N., Stasse, O.: MonoSLAM: Real-time single camera SLAM. PAMI (2007)
31. Smith, P., Reid, I., Davison, A.J.: Real-time monocular slam with straight lines. In: BMVC. (2006)
32. Hirose, K., Saito, H.: Fast line description for line-based slam. In: BMVC. (2012)
33. Holmes, S.A., Murray, D.W.: Monocular SLAM with conditionally independent split mapping. PAMI (2013)
34. Strasdat, H., Montiel, J.M.M., Davison, A.: Real-time monocular SLAM: Why filter? In: ICRA. (2010)
35. Klein, G., Murray, D.: Parallel tracking and mapping for small AR workspaces. In: ISMAR. (2007)
36. Newcombe, R.A., Davison, A.J.: Live dense reconstruction with a single moving camera. In: CVPR. (2010)
37. Newcombe, R.A., Lovegrove, S., Davison, A.: DTAM: Dense tracking and mapping in real-time. In: ICCV. (2011)
38. Zhang, L.: Line Primitives and Their Applications in Geometric Computer Vision. PhD thesis, Department of Computer Science, Kiel University (2013)
39. Vedaldi, A., Fulkerson, B.: VLFeat: An open and portable library of computer vision algorithms. `http://www.vlfeat.org/` (2008)
40. Lebeda, K., Matas, J., Chum, O.: Fixing the locally optimized RANSAC. In: BMVC. (2012)
41. Chum, O., Matas, J.: Matching with PROSAC - PROgressive SAmple Consensus. In: CVPR. (2005)
42. von Gioi, R., Jakubowicz, J., Morel, J.M., Randall, G.: LSD: A fast line segment detector with a false detection control. PAMI (2010)
43. Hartley, R.I., Zisserman, A.: Multiple View Geometry in Computer Vision. Second edn. Cambridge University Press (2004)
44. Agarwal, S., Mierle, K., Others: Ceres solver. (`http://code.google.com/p/ceres-solver/`)
45. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning. MIT Press (2006)
46. Hensman, J., Fusi, N., Andrade, R., Durrande, N., Saul, A., Zwiessele, M., Lawrence, N.D.: GPy library. (`http://github.com/SheffieldML/GPy`)
47. Wu, C., Agarwal, S., Curless, B., Seitz, S.M.: Multicore bundle adjustment. In: CVPR. (2011)
48. Wu, C.: Towards linear-time incremental structure from motion. In: 3DV. (2013)
49. Snavely, N., Seitz, S.M., Szeliski, R.: Photo tourism: Exploring image collections in 3D. In: SIGGRAPH. (2006)
50. Chen, M., Pang, S.K., Cham, T.J., Goh, A.: Visual tracking with generative template model based on riemannian manifold of covariances. In: ICIF. (2011)