TMAGIC: A Model-free 3D Tracker

Karel Lebeda, Simon Hadfield, *Member, IEEE,* and Richard Bowden, *Senior Member, IEEE*

***Abstract*—Significant effort has been devoted within the visual tracking community to rapid learning of object properties on the fly. However, state-of-the-art approaches still often fail in cases such as rapid out-of-plane rotation, when the appearance changes suddenly. One of the major contributions of this work is a radical rethinking of the traditional wisdom of modelling 3D motion as appearance change during tracking. Instead, 3D motion is modelled as 3D motion. This intuitive but previously unexplored approach provides new possibilities in visual tracking research.**

Firstly, 3D tracking is more general, as large out-of-plane motion is often fatal for 2D trackers, but helps 3D trackers to build better models. Secondly, the tracker's internal model of the object can be used in many different applications and it could even become the main motivation, with tracking supporting reconstruction rather than vice versa. This effectively bridges the gap between visual tracking and Structure from Motion.

A new benchmark dataset of sequences with extreme out-of-plane rotation is presented and an online leader-board offered to stimulate new research in the relatively underdeveloped area of 3D tracking. The proposed method, provided as a baseline, is capable of successfully tracking these sequences, all of which pose a considerable challenge to 2D trackers (error reduced by 46 %).

I. Introduction

ONE of the major challenges found in visual tracking is *out-of-plane rotation*, caused by variations in viewpoint. This is a very hard problem, causing failures of many state-of-the-art trackers due to the radical change in appearance that can ensue. It is obvious that any successful tracker needs to address this kind of appearance change, which comes from the object pose and cannot be modelled by a simple planar transformation.

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 [10], [28], a cloud of 2D points [4], [36] or a combination of these [14]. Unfortunately, variations of viewpoint lead to rapid changes in 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.

In this work, the conventional approach of treating appearance changes resulting from viewpoint variation as object diversity is challenged. Instead, it is argued that an intrinsically 3D object in the 3D world should be modelled as such. Variations of viewpoint (relative camera-object motion) should then be treated explicitly as the camera motion, in accordance with reality. Following this path, the negative effects of out-of-plane rotation are not only mitigated; they actually prove beneficial, as they improve the numerical conditioning (wider baseline).

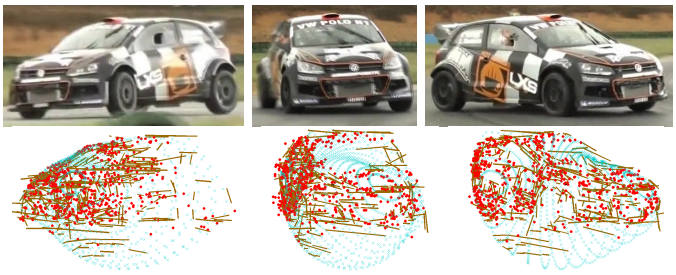

Fig. 1. Out-of-plane rotations change the object appearance significantly, here is a complete change in just 50 frames. First row: original images. Second row: feature cloud and final model returned by the tracker. Notice the bottom and back side of the car, which have not been observed yet, so the point cloud does not reach there and the model is smoothly extrapolated.

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 approach 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 common in SLAM [21]. Another difference from SfM/SLAM is object/background segmentation, where only a small portion (*e.g.* 10 %, but can be even as low as 1 %) of the image may be used in tracking. This can easily become a significant issue as an SfM/SLAM technique would attempt to model the scene (background) while features on the object would be rejected as outliers. However, exactly the opposite behaviour is desired. Therefore it is vital to use the tracking to provide object/background segmentation to focus on the target object and *actively ignore* the background.

To achieve this, a novel approach to modelling the object 3D shape using a Gaussian Processes (GP) is explored. This model helps to distinguish (segment) which parts of the image belong to the projection of the object and which are background, allowing intelligent detection of new features, thus preventing inclusion of the background in the model. 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 that would lead to failure of techniques such as PTAM [16], 3) the surface normals of the GP indicate which points on the object may be visible to a particular camera to aid redetection and loop closure and 4) the GP offers a model of the surface for visualisation and subsequent tasks, such as dense reconstruction or robot navigation, etc.

K. Lebeda (corresponding author), S. Hadfield and R. Bowden are with University of Surrey, GU2 7XH Guildford, United Kingdom (e-mail: *karel@lebeda.sk,* {*s.hadfield,r.bowden*}*@surrey.ac.uk*).

This work was supported by the EPSRC project EP/I011811/1 and the SNSF project SMILE. The authors would like to thank to Dr Philip Krejov for his help with creating synthetic data.

Manuscript created February 13, 2017.

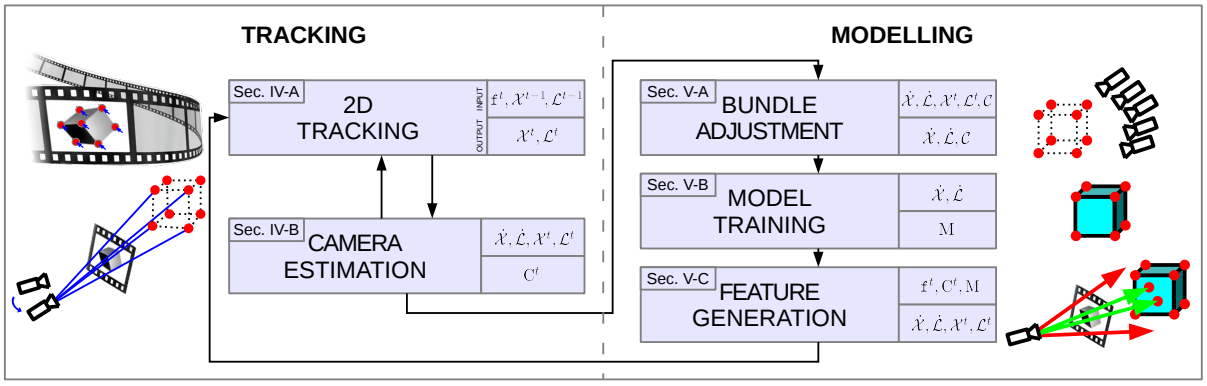

Fig. 2. Overview of the TMAGIC tracker. Symbols on the right side of each step indicate inputs and outputs, as labelled for the case of 2D tracking. While the tracking loop is repeated in every frame, modelling only runs when necessary (according to Equation (4)). The numbers in upper left corners denote sections covering individual steps.

This article expands our previous conference publication [18]. To aid reproducibility we include additional insight into the internal workings of the algorithm, especially the feature management and the Gaussian Process modelling. This includes new figures and formalisation. Furthermore, experimental evaluation is greatly expanded. Finally, the new sequences which form the extended evaluation of the technique are made available online including ground truth. Since very few works exist on model-free 3D tracking, a benchmark with an online leader-board is set up to stimulate new research in the area[1].

The rest of this article is structured as follows. After discussing related work in Section II, the TMAGIC tracking algorithm is introduced in Section III. Its workings are detailed in Sections IV and V regarding the tracking and modelling parts, respectively. Section VI brings the experimental evaluation of the algorithm from the point of view of 2D tracking, 3D tracking and 3D modelling. Finally, Section VII draws conclusions and discusses limitations of the proposed approach.

II. Related Work

3D monocular tracking attempts to recover the trajectory of the object in the 3D world (relative to the camera) instead of in the image plane. Typically, techniques employ known 3D models of the object, examples of which are the tracking of the pose of human bodies [29], vehicles in traffic scenes [8], [38] or general *boxes* [15].

The focus of this work is on model-free tracking. There has been considerably less research in this field. While there have been relatively few attempts at learning 3D tracking models on the fly, such approaches are fundamental to online SfM (such as [13] or [24], where online coarse reconstruction is provided as feedback for guided scanning). However, such approaches assume the interest is in reconstructing the whole scene. In the area of visual object tracking, 3D based approaches must extract 3D shape and trajectory when objects are represented only by a minority of the video frame. For this reason, motion segmentation is sometimes used, such as to track and reconstruct moving bodies in the SLAM pipeline of Kundu *et al.* [17].

An example of a recent model-less 3D tracker is the work of Feng *et al.* [9], who introduced the idea of 3D monocular tracking with no offline modelling or training, using explicit colour-based segmentation. This allowed reconstruction of the segmented object using visual SLAM techniques. However, they do not attempt to estimate the surface shape beyond a cloud of points. Another similar approach is the work of Prisacariu *et al.* [26] who use level-set techniques for 3D modelling. The approach is however not online, as it processes the video-sequence as a batch, which significantly limits possible applications of such a "tracker". It is important to mention the tracker of Yin and Collins [41], where the camera pose is estimated in 3D during 2D object tracking, without building any 3D model. As such, this work can be seen as an intermediate step between 2D and 3D approaches.

There has been a significant amount of work in the area of online 3D modelling based on depth sensors. The most notable in this area is the work of Newcombe *et al.*, in particular Kinect Fusion [23] and Dynamic Fusion [22]. However, since this work focuses on tracking and modelling from strictly monocular RGB video, a more detailed review is omitted here. Recommended recent overviews for 3D modelling and tracking are [5] and [33], respectively.

III. Simultaneous 3D Tracking and Reconstruction

Our goal is the tracking of a 3D object throughout a sequence, learning a model of appearance on the fly. However, unlike most online tracking approaches, ideas from both SfM and SLAM are employed in this work, to form a 3D representation of the model that can cope with out-of-plane rotation. The program and data flow of the proposed TMAGIC (standing for Tracking, Modelling and Gaussian-Process Inference Combined) tracker is illustrated in Figure 2. It consists of two parts: *tracking* and *modelling* (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 IV-A), yielding the sets of features currently

[1] http://cvssp.org/data/3DCars/

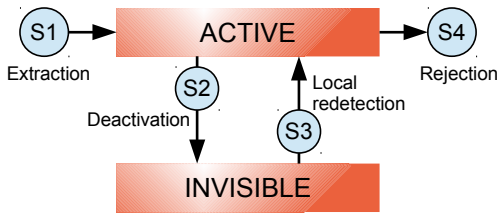

Fig. 3. Life cycle of features used for 3D tracking.

visible. Using these, the new camera pose can then be estimated (Section IV-B), while keeping the corresponding 3D features (points and lines in the real world) fixed. Without an outer reference, the world coordinate system is not fixed. Therefore it can be, without loss of generality, safely assumed 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 BA (Section V-A). 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 V-B), which can be exploited in two ways. The model defines regions of the image which are eligible to detect new 2D features. Secondly, it provides an initialisation of the corresponding backprojected 3D features (Section V-C). Features, successfully extracted using the current frame, camera pose and the shape model, enrich the 2D and 3D sets for use in future tracking.

IV. Camera Pose Tracking

A. 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 [42], *e.g.* constraining the orientation of the surface. They can be not only texture-based, but also stemming from the shape of the object [19]. Therefore in man-made environments they appear in situations where point features are scarce [30] (such as in a low texture scenario).

Features of both types go through the same life cycle, visualised in Figure 3. Firstly, they are extracted from the image, in areas belonging to (*i.e.* segmented as) the tracked object. This is denoted as S1. Newly created features are regarded as *active*. These features may be deactivated (and their 2D counterparts removed), if they cannot be tracked, or are deemed to be on an invisible part of the object. This transition to the *invisible* state is marked as S2. On the other hand, *invisible* features may be redetected and thus return to the *active* state (S3). Finally, features considered invalid (*e.g.* laying on the background) are discarded (S4). The techniques used are described in the following paragraphs and the role of the online learned model is detailed in Section V-C.

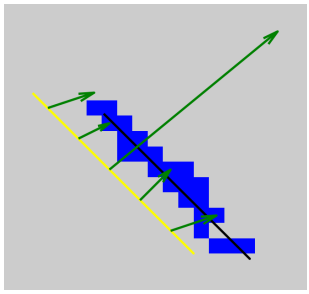

Fig. 4. An example of line tracking. Yellow: line feature $\mathbf{l}_i^{t-1}$, black: line feature $\mathbf{l}_j'^t$, blue: pixels belonging to $\mathbf{l}_j'^t$ (as given by LSD), green: edge-to-edge correspondences. Since 3 out of the 5 sampled edge points converged, feature $\mathbf{l}_j'^t$ is validated and will become $\mathbf{l}_i^t$.

The 2D point features $\mathbf{x}_i^t \in \mathcal{X}^t$ are extracted (S1) using two techniques: Difference of Gaussians (*i.e.* SIFT points) and Hessian Laplace [35]. These features are tracked independently by an LK tracker from frame $t-1$ to t. Features, which do not converge are removed from the 2D feature cloud (S2). The tracked features generate a set of correspondences $\{(\mathbf{x}_i^{t-1}, \mathbf{x}_i^t)\}$, which are subsequently verified by LO-RANSAC [7], [20]. Outliers to RANSAC, *i.e.* correspondences inconsistent with a global epipolar geometry model, are removed (S4), as well as their respective 3D features, as these are likely to lie on the background.

The 2D line features $\mathbf{l}_i^t \in \mathcal{L}^t$ are extracted (S1) using the LSD [37] approach with false-positive detection control. Lines are tracked as follows. Firstly, the LSD is executed on frame t to obtain a set of candidate segments $\mathbf{l}_j'^t \in \mathcal{L}'^t$. Along each of the previous line segments $\mathbf{l}_i^{t-1}$ a number of edge points are then sampled. Each of these is tracked independently, using the *guided edge search* of [19], leading to a new edge point in the current frame. If this new point belongs to a line segment in $\mathcal{L}'^t$, this point votes for the segment. The segment $\mathbf{l}_j'^t$ with the most votes becomes the new feature $\mathbf{l}_i^t$. As there is no way to validate line correspondences w.r.t. an epipolar geometry [11], the features are validated using a threshold on the minimum number of votes (3 out of 5 in all the experiments). See Figure 4 for an example.

B. Camera Estimation

The camera C^t 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

$$\mathtt{P}^t = \frac{1}{f}\mathtt{K}\mathtt{R}^t \left[\mathtt{I}_3 | -\mathbf{C}^t\right] , \tag{1}$$

where f is a focal length (in world units, e.g. millimetres) and $\mathtt{K}$ is a calibration matrix of intrinsic camera parameters [11], while $\mathtt{I}_3$ stands for the 3×3 identity matrix. Since $\mathtt{P}^t$ is a homogeneous entity, f can be neglected in the computation. For simplicity, a general projection function Π is defined, such that 3D lines are projected as $\mathbf{l}_i^t = \Pi(\mathbf{L}_i|\mathrm{C}^t)$ and 3D points as $\mathbf{x}_i^t = \Pi(\mathbf{X}_i|\mathrm{C}^t)$.

Assuming a cloud of 3D features (points $\mathbf{X}_i \in \dot{\mathcal{X}}$ and line segments $\mathbf{L}_i \in \dot{\mathcal{L}}$, which are defined by their end-points) and their projections ($\mathcal{X}^t$, $\mathcal{L}^t$) is given, it is possible to estimate a

pose (in particular $\mathtt{R}^t, \mathbf{C}^t$) of the camera C^t. The calibration matrix $\mathtt{K}$ of the camera is not computed exactly, instead an estimate based on image dimensions is used:

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

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*. [25] 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 (the so-called PnP problem) is a standard problem, *e.g*. solved by a P3P-RANSAC [11]. However, although research has been done in the P3L/PnL area for lines [42], combining these two is not straightforward. Therefore an optimisation approach is used to solve for camera pose:

$$\begin{aligned} \mathrm{C}^t = \arg\min_{\underline{\mathrm{C}}} & \sum_{i=1}^{|\mathcal{X}^t|} \rho_1 \left(||\mathbf{x}_i^t - \Pi(\mathbf{X}_i|\underline{\mathrm{C}})|| \right) + \\ & \sum_{i=1}^{|\mathcal{L}^t|} \left(\rho_1 \left(\widetilde{\boldsymbol{\mu}}_{\mathbf{l}_i^t}^\top \Pi(\mathbf{L}_i|\underline{\mathrm{C}}) \right) + \rho_1 \left(\widetilde{\boldsymbol{\nu}}_{\mathbf{l}_i^t}^\top \Pi(\mathbf{L}_i|\underline{\mathrm{C}}) \right) \right), \end{aligned} \tag{3}$$

using both point and line features in a unified framework and exploiting the sequential nature of tracking; ρ_1 is a robust cost function to provide outlier tolerance (similar to [34]). 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 reprojection error. For line features, the error terms are defined as orthogonal distances of the endpoints of the segment $\mathbf{l}_i$ ($\widetilde{\boldsymbol{\mu}}_{\mathbf{l}_i}, \widetilde{\boldsymbol{\nu}}_{\mathbf{l}_i}$, in homogeneous coordinates), to the projection of the 3D line $\Pi(\mathbf{L}_i|\underline{\mathrm{C}})$ (homogeneous, normalised to the unit length of the normal vector). Note that since the line may not be fully visible (and is theoretically infinite), only perpendicular distances are used, in order to cope with the *aperture problem* [19], [30]. This minimisation is initialised at the pose in the previous frame (C^{t-1}). During experimentation, it was found that due to the smooth nature of the derivatives of (3), the basin of convergence for this optimisation is several orders of magnitude larger than the typical inter-frame difference.

In the first frame, the world coordinate frame (which is defined only up to a similarity) is chosen as follows. The object, initialised as a sphere (see Section V-B for more details), is centred at the origin and the 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 9). This is, however, not fixed, and changes freely during BA, which optimises both the camera trajectory and feature positions.

V. Online Modelling 3D Shapes

A. Bundle Adjustment

After initialisation, 2D tracking is performed until the distance between camera centres exceeds a specified threshold $\theta_{\mathbf{C}}$ [24]:

$$||\mathbf{C}^t - \mathbf{C}^{t'}|| > \theta_{\mathbf{C}}, \tag{4}$$

where t' is the time of the last BA. Theoretically, $\theta_{\mathbf{C}}$ could be set to 0 such that BA is executed on every frame, however that would be excessively time-consuming. Requiring a baseline of sufficient width (non-negligible camera motion) between two consecutive BA runs creates well-timed on-demand executions on keyframes characterized by equidistant camera poses. When the condition (4) is satisfied, the modelling part of the algorithm is performed. Firstly, the Bundle Adjustment refines the positions of 3D features $\dot{\mathcal{X}}$, $\dot{\mathcal{L}}$ and cameras $\mathcal{C}$ (for the purposes of BA, we define $\mathcal{C}$ as the set of previous cameras $\mathrm{C}^1,\ldots,\mathrm{C}^t$). If speed is an issue, one may limit the BA to take only the last k cameras into account, *i.e*. to define $\mathcal{C} = \{\mathrm{C}^u | u = \max(1, t-k), \ldots, t\}$, however, in experiments within this work this did not prove necessary (thus the setting $k = \infty$ was employed). The BA [2] minimises a similar error to (3):

$$\begin{aligned} \arg\min_{\dot{\mathcal{X}},\dot{\mathcal{L}},\mathcal{C}} \sum_{u=1}^{t} \Bigg(& \sum_{i=1}^{|\mathcal{X}^u|} \rho_2 \left(||\mathbf{x}_i^u - \Pi(\mathbf{X}_i|\mathrm{C}^u)|| \right) + \\ & \sum_{i=1}^{|\mathcal{L}^u|} \left(\rho_2 \left(\widetilde{\boldsymbol{\mu}}_{\mathbf{l}_i^u}^\top \Pi(\mathbf{L}_i|\mathrm{C}^u) \right) + \rho_2 \left(\widetilde{\boldsymbol{\nu}}_{\mathbf{l}_i^u}^\top \Pi(\mathbf{L}_i|\mathrm{C}^u) \right) + \Lambda_i \right) \Bigg), \end{aligned} \tag{5}$$

where the additional term Λ_i is a regularisation term, which ensures that the lengths of 3D line segments are close to those observed. The robust cost function ρ_2 employed here is the Cauchy loss, as provided by the Ceres Solver [2]. Note also that every point from $\mathcal{X}^t$ and $\mathcal{L}^t$ has a correspondence in $\dot{\mathcal{X}}$ and $\dot{\mathcal{L}}$, respectively, for every t, but not necessarily vice-versa, due to features which are not currently visible. After the 3D feature positions have been refined, they are used to train the shape model.

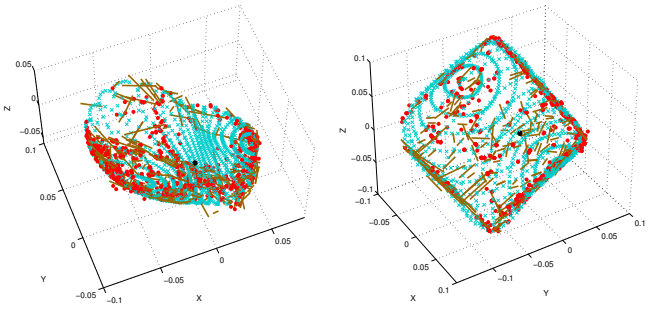

Fig. 5. Examples of 3D feature clouds and GP-learned smooth models. Notice the unseen parts of the objects, which are without features and where the model is extrapolated (*i.e*. the rear side of the car and the "pole" on the left side of the cube).

B. Gaussian Process Modelling

As discussed previously, the object shape is modelled as a Gaussian Process (GP) [1], [27]. This provides the ability to infer a fully dense 3D model from the finite collection of discrete observations $\dot{\mathcal{X}}$ and $\dot{\mathcal{L}}$. There are usually hundreds of features, however in cases of low-resolution video (*e.g*. `Dog`)

there can be less than a hundred training points. In theory it is possible to learn the hyperparameters from less than 10 features, however this leads to low accuracy. 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.

For online target/background segmentation, a GP is trained as a coarse, probabilistic model. A representation is chosen such that every point $\mathbf{X}$ on the surface is represented in spherical coordinates (θ, φ) relative to the object centre $\mathbf{Y}_\circ$ as:

$$\mathbf{X} = \mathbf{Y}_\circ + r \cdot (\sin\theta\, \cos\varphi, \sin\theta\, \sin\varphi, \cos\theta)^\top \tag{6}$$

(for more details on the choice of $\mathbf{Y}_\circ$ see below). For any pair of angles, the radius would then be modelled as:

$$r = \mathrm{GP}(\theta, \varphi | \kappa)\,, \tag{7}$$

where κ is the *kernel* of the GP, relating to the surface properties of the modelled object, and may be any positive definite two-parameter function (see below for more details). This function is learned during tracking (per sequence), such that the likelihood of the training data is maximised.

This is a natural, minimal (*i.e.* using exactly 2 parameters for the two-dimensional space of 3D directions) parameterisation. It does, however, suffer from singularities at the "poles", where the whole range of azimuths represent a single point. For this reason, the unit-vector parameterisation is used within this work instead. A 3D point is first re-expressed as a vector from the centre $\mathbf{Y}_\circ$ of the object:

$$\mathbf{X} = \mathbf{Y}_\circ + \mathbf{Y} \tag{8}$$

These are then modelled such that the unit-length normalised vectors $\bar{\mathbf{Y}} = \mathbf{Y}/||\mathbf{Y}||$ represent the independent variable and the radii $r_{\bar{\mathbf{Y}}} = ||\mathbf{Y}||$ the dependent variable, *i.e.*:

$$\mathbf{X} = \mathbf{Y}_\circ + r \cdot \bar{\mathbf{Y}}\,, \tag{9}$$

$$r = \mathrm{GP}(\bar{\mathbf{Y}} | \kappa)\,. \tag{10}$$

As it does not suffer from a singularity in any direction, this parameterisation was found superior to alternatives such as spherical coordinates, despite its higher dimensionality.

The observed 3D points $\dot{\mathcal{Y}} = \{\mathbf{Y}_i\}$ (point features $\dot{\mathcal{X}}$ and end-points[2] of line features $\dot{\mathcal{L}}$, in both the *active* and *invisible* states) are used as training data for the Gaussian Process (GP). It could be argued that the 3D parameter space in this new representation is not sufficiently covered by training data. It is true that only training points laying on the unit sphere (a 2D manifold) are provided and the parameter space outside the manifold is unconstrained. However, this does not create a problem in practice, since only query points laying on the unit sphere (*i.e.* direction vectors) are queried.

Without loss of generality, we can assume that the centre $\mathbf{Y}_\circ$ coincides with the origin of the world coordinate system.

[2]It is possible to sample more points along lines which have high confidence.

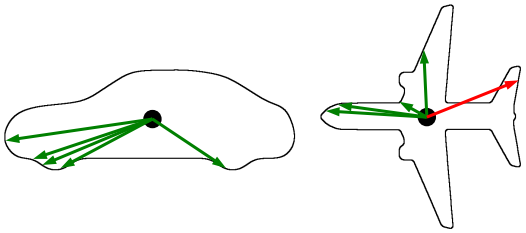

Fig. 6. Star domain example in 2D. Left: Every region of the car shape can be reached from the *centre* without crossing the boundary, *i.e.* its shape is a star domain. Right: Since there are regions unreachable by a straight line, it is not a star domain.

In this case, for a query direction $\bar{\mathbf{Q}}$ (where $||\bar{\mathbf{Q}}|| = 1$), the resulting 3D point $\mathbf{Q}$ is predicted as [27]:

$$\mathbf{Q} = \bar{\mathbf{Q}} \quad \begin{pmatrix} \kappa(\dot{\bar{\mathcal{Y}}}, \bar{\mathbf{Q}})^\top\ \kappa(\dot{\bar{\mathcal{Y}}}, \dot{\bar{\mathcal{Y}}})^{-1}\ \mathbf{r}_{\dot{\bar{\mathcal{Y}}}} \\ \pm\kappa(\dot{\bar{\mathcal{Y}}}, \bar{\mathbf{Q}})^\top\ \kappa(\dot{\bar{\mathcal{Y}}}, \dot{\bar{\mathcal{Y}}})^{-1}\ \kappa(\dot{\bar{\mathcal{Y}}}, \bar{\mathbf{Q}}) \end{pmatrix}, \tag{11}$$

or more succinctly as

$$\mathbf{Q} = \bar{\mathbf{Q}}\ \left(r_{\bar{\mathbf{Q}}} \pm \sigma_{\bar{\mathbf{Q}}}\right)\,, \tag{12}$$

where $r_{\bar{\mathbf{Q}}}$ is the predicted radius and $\sigma_{\bar{\mathbf{Q}}}$ is the confidence. The notation $\mathbf{r}_{\dot{\bar{\mathcal{Y}}}}$ represents a vector of norms of all vectors in the training set $\dot{\bar{\mathcal{Y}}}$, *i.e.* $\mathbf{r}_{\dot{\bar{\mathcal{Y}}};i} = r_{\bar{\mathbf{Y}}_i} = ||\mathbf{Y}_i||$.

Intuitively, Equation (11) 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 influence of any particular element of the training data is quantified by $\kappa(\dot{\bar{\mathcal{Y}}}, \bar{\mathbf{Q}})$, while $\kappa(\dot{\bar{\mathcal{Y}}}, \dot{\bar{\mathcal{Y}}})^{-1}$ removes any correlation within the training data.

This kind of representation means that the object shape is modelled by an *implicit non-parametric function*. While one can query the surface in any direction, there is no discrete "set of vertices" marking the shape. Instead, for visualisation the model is queried at regularly sampled positions (see Figure 5 for an example). This, however, is not an obstacle for its use in the presented framework.

As mentioned previously, the Gaussian Process shape model is fully probabilistic. That means that not only a shape estimate is provided, but a whole distribution of shapes (radius functions). From this distribution, the mean (*i.e.* the most probable) shape is used as the estimate, and the variance as the uncertainty at any given point of the object surface. The probabilistic nature of the GP model furthermore prevents overfitting through an implicit "Occam's-razor" effect, that favours models which are both simple and which explain the observations well. Other beneficial properties of GP modelling include smooth interpolation and extrapolation in regions without training inputs. Several alternatives for the surface shape modelling were tested, such as a mesh-based model, a parametric probability distribution and a spherical-coordinate model with a different machine-learning technique used (such as nearest neighbour regression, neural network, random regression forest or support vector regression). However, none had the properties required.

The non-parametric nature of the model used makes it possible to model a wide range of object shapes and resolutions without the need for reparameterisation. To specify rigorously what class of objects can be modelled, one needs to consider

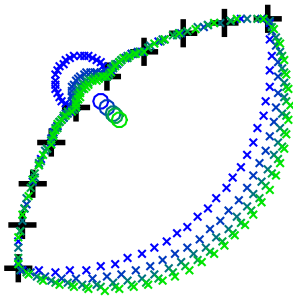

Fig. 7. Iterative search for the shape centre. Black: training data. Coloured: shape centre (○) and sampled points (×), iterating from the centre of mass $\overline{\mathbf{X}}$ (blue) to convergence (green).

the properties of the spherical representation. Since the radius for any given direction angle must be unique, there must exist a point inside the object, the *shape centre*, such that the line segments connecting it to all the points on the shape surface lie inside the object. This class of objects is known in computational geometry as *star shapes* or *star domains* (of a Euclidean space). See Figure 6 for 2D examples of objects which are and are not a star domain. While this choice of parameterisation might seem overly limiting, in practice most "compact" objects (without deep concavities or long, extended parts) are of approximately star shape. Furthermore, it is not necessarily harmful for a particular application of the modelling (such as segmentation in visual tracking) when the online model smooths over minor regions which break this assumption.

Attention must be paid to the selection of the *shape centre* $\mathbf{Y}_\circ$. While using simply the centre of mass is a viable solution for many shapes, sometimes it lies too close to the object surface, which leads to unwanted artefacts (see the blue samples in Figure 7). Therefore a data-driven shape centre is found as follows. The centre of mass of the training points is used only as an initialisation and the centre is subsequently shifted towards the midpoint between this and the centre of mass of the sampled points (trained with the previous centre). The sampled points change with the shift of the centre, so this needs to be iterated:

$$\mathbf{Y}_\circ^{\text{new}} = \lambda \, \frac{\frac{1}{|\dot{\mathcal{Y}}|} \sum_{\dot{\mathcal{Y}}} \mathbf{Y}_i + \frac{1}{|\dot{\mathcal{M}}|} \sum_{\dot{\mathcal{M}}} \mathbf{M}_i}{2} + (1-\lambda)\mathbf{Y}_\circ \quad (13)$$

where λ is a learning factor (set to 0.5 in our experiments) and $\mathbf{M}_i$ is a point sampled on the surface model (in regular angular intervals, visualised as ×). See Figure 7 for a 2D illustration of the convergence. It is necessary to repeat this search for the shape centre every time the training data changes (after every bundle adjustment). However, it is *not* necessary to repeat all steps to full convergence. Instead, only one step is performed after each bundle adjustment, which converges eventually as the (relative) magnitude of updates of the training data decreases.

One of the most important choices while designing a technique using a GP is the choice of kernel (or combination of kernels). The kernel choice represents prior knowledge about properties of the modelled function (in this case surface shape), especially smoothness and differentiability. One of the most commonly used is the RBF kernel (also called Gaussian kernel), which is infinitely differentiable and therefore induces smooth

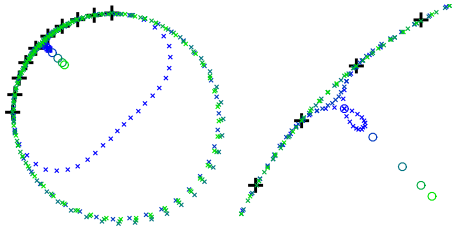

Fig. 8. Iterative search for the shape centre with the RBF kernel; legend is the same as in Figure 7. Right: detail of regression near the shape centre.

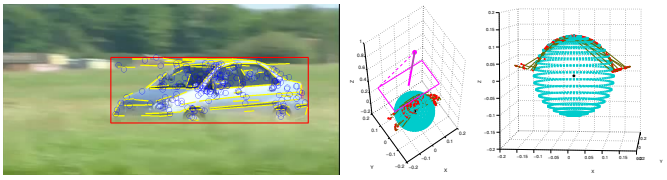

Fig. 9. State of the proposed TMAGIC tracker after the first frame. Left: a bounding box, $\mathcal{X}^1$ and $\mathcal{L}^1$. Right: $\dot{\mathcal{X}}$, $\dot{\mathcal{L}}$, initial model M (the blue sphere visualises sampled points $\dot{\mathcal{M}}$) and C^1 (magenta). For the camera, the visualisation 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.

shape contours. This, however, causes artefacts in the shape, as the overly smooth gradient extrapolates too far from the training data and exaggerates rapid radius changes (see Figure 8). The same holds for the Matérn kernel (of both common orders 3/2 and 5/2). For this reason, the *exponential* kernel is employed, which allows fast changes in both the radius and its gradient (which can even be discontinuous) and thus models sharp edges. This kernel is (additively) combined with a *bias* kernel to avoid the assumption of zero-centred data and with a *white-noise* kernel to gain robustness against outliers:

$$\kappa_{\text{GP}} = \kappa_{\text{Exp}} + \kappa_{\text{B}} + \kappa_{\text{W}} \,. \quad (14)$$

All the kernel parameters (including the width of the exponential kernel and the sub-kernel weights) are learned from the data at the training stage. Besides the choice of kernel, there are no other parameters in the GP modelling.

It should be noted, that at the beginning of the video-sequence (*i.e.* before the first Bundle Adjustment), there is no depth estimate and therefore no 3D information. As an initialisation of the model, a sphere is used, with dimensions infered from the initial bounding box provided to the tracker in the first frame (see Figure 9). While this model would be useless as an output, it provides a prior to initialise tracking and the model is then trained as soon as the 3D feature positions are refined.

C. Feature generation

For camera pose estimation (Sec. IV-B), it was assumed that the 3D feature clouds $\dot{\mathcal{X}}$ and $\dot{\mathcal{L}}$ are known and fixed. In this section, the issue of feature generation and localisation is addressed. The assumption is made that a shape model of the object is given (trained according to the previous section). The model is vital at this stage, as it effectively segments out the object from the background, validating features for use within the tracker. This however does not hold for the first frame, see below.

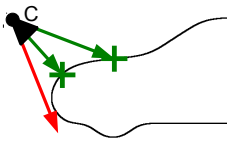

Fig. 10. Model intersection search example. The intersection points (+) are initial 3D locations of the newly generated features. $\mathbf{C}$ marks the camera centre.

In the first frame, initial sets of 2D features $\mathcal{X}^1$ and $\mathcal{L}^1$ are generated inside a user-supplied bounding box. When generating a 3D feature $\mathbf{X}_i$ for a new 2D feature $\mathbf{x}_i^t$ (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 α) from the camera centre is generated:

$$\mathbf{Z}(\alpha) = \mathbf{C}^t + \alpha(\mathtt{K}\mathtt{R}^t)^{-1}\widetilde{\mathbf{x}}_i^t \qquad (\alpha > 0)\,, \tag{15}$$

where $\widetilde{\mathbf{x}}_i^t$ is the homogeneous representation of $\mathbf{x}_i^t$. Then a search for the (nearest to the camera) intersection between the ray and the shape is performed:

$$\mathbf{X}_i = \mathbf{Z}(\alpha)\,, \tag{16}$$

$$\alpha = \underset{\underline{\alpha}>0}{\arg\min}\,\underline{\alpha} \qquad \text{s.t.} \quad ||\mathbf{Z}(\underline{\alpha})|| = r_{\bar{\mathbf{Z}}(\underline{\alpha})}\,. \tag{17}$$

If the minimisation of (17) has no solution, it means that the ray does not intersect the *mean surface* given by the GP (from the distribution of possible surfaces, see Figure 10). 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 $\sigma_{\bar{\mathbf{Z}}(\underline{\alpha})}$) to $r_{\bar{\mathbf{Z}}(\underline{\alpha})}$ in Equation (17).

It should be noted that as in parts of the previous section, the shape centre $\mathbf{Y}_\circ$ is assumed to be the origin of the coordinate system, to keep the notation uncluttered. Therefore the (finite-length) ray can be simply expressed, similarly to Equation (12), as cast from the origin: $\mathbf{Z} = r_{\bar{\mathbf{Z}}} \cdot \bar{\mathbf{Z}}$. This, again, does not limit generality, as reintroducing $\mathbf{Y}_\circ$ back to the equations is trivial.

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 $t > 1$, it is not used: features are detected over the whole image but only those covering the target object are used. However, in the first frame, all the 2D features will lie inside the user-specified bounding box. In this case, their 3D positions are reconstructed such that they minimise the distance to the mean surface (even when they do not intersect), leading to the characteristic fringe seen in Figure 9:

$$\alpha = \underset{\underline{\alpha}>0}{\arg\min}(||\mathbf{Z}(\underline{\alpha})|| - r_{\bar{\mathbf{Z}}(\underline{\alpha})})^2\,. \tag{18}$$

Generation of new features in $t > 1$ 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:

$$\sigma_{\bar{\mathbf{Z}}(\alpha)} > \theta_\sigma\,. \tag{19}$$

As previously mentioned, some of the 3D features may be temporarily occluded, *i.e.* without 2D correspondences (in the *invisible* state). Surface normals, given by the shape model, provide a tool to determine which parts of the object are visible from a particular direction. This is done by sampling three points in close proximity to the location of interest and locally fitting a tangent plane. To remove any effect of shape curvature, the distance is several orders of magnitude below the object dimensions. TMAGIC uses this information in two complementary ways. Firstly, if a 3D feature is deemed not visible, but it has a 2D correspondence (e.g. adheres to an object contour), it can be removed (transition S2 in Figure 3). On the other hand, if a 3D feature has no 2D correspondence, but is on a surface which is seen by C^t under an angle close to normal, the 2D feature can be redetected (S3). This is performed by projecting it into frame t by Π 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.

VI. Experimental Evaluation

In all experiments, the parameters were fixed as follows: $\theta_{\mathbf{C}} = 10\,\%$, $\theta_\sigma = 0.5\,\%$, relative to the scene size. The tested proof-of-concept implementation (Matlab framework with several parts in C++) currently runs at speeds between 1 and 3 s per frame on an Intel i5 computer at 3.3 GHz and takes around 1 GB of RAM. However, there are possibilities for trivial technical improvements and for parallelisation, allowing significantly higher speeds. For instance, processing of point and line features is independent from each other and could be executed in separate threads; similarly the modelling loop can be offloaded onto an independent semi-synchronous thread, similarly to *e.g.* PTAM [16]. Independently to these, the optimisation in both BA and model training (including inversion of large matrices) offers possibility for massive parallelisation, both multi-CPU and GPU.

A. Synthetic data

Firstly, results on a synthetic sequence CUBE1 are shown (Figure 11). 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 the TMAGIC tracker, the camera circles around the fixed cube with a perfect circular trajectory (see Figure 12). However, since the 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 a 3D circle is fitted to the camera trajectory and the error measured as an orthogonal distance from this circle. Figure 13 shows 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.

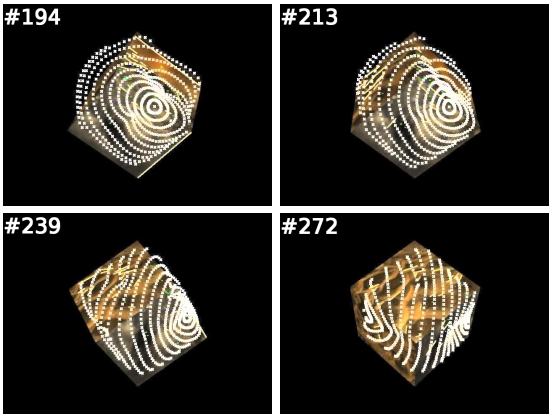

Fig. 11. Selected frames from the CUBE1 sequence. Notice, how TMAGIC learns the new face of the cube. #194: unknown shape, the surface is smoothed over. #213: first features detected, shape roughly estimated. #239: more features identified, shape refined. #272: Final state, model in agreement with the object.

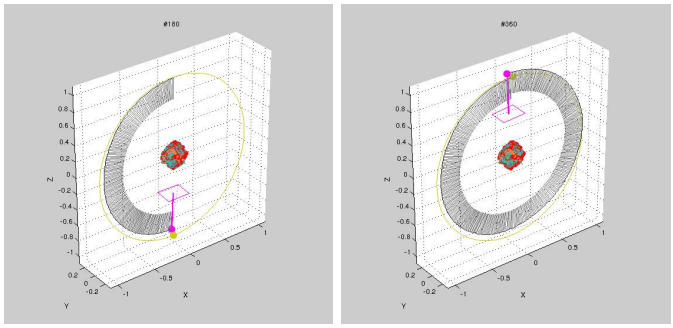

Fig. 12. The 3D scene in $t = 180$ and 360. The camera and features are shown in the same way as in Figure 9, the camera trajectory (centres and principal directions for each previous frame) is shown in black. Ground-truth trajectory is in yellow. The details of the model can be seen in Figure 5.

Despite being based on sparse data, the resulting learned shape model represents the cubic shape (of side equal to approximately 17 cm) accurately, having a mean reconstruction error of 3.4 mm. 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. The input video and additional results can be found on the authors' website.

To compare the method with currently used reconstruction approaches, this sequence (with no background to account for) was processed with VisualSfM [39], [40], Bundler [31], [32] and CMP SfM WebService [12]. 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. On the other hand, CMP SfM reconstructed the scene with a lower camera trajectory error of 0.7 cm, but higher model error of 7.0 mm. On real sequences, including background, these reconstruction techniques perform even worse, commonly failing to capture the object of interest (see Figure 14).

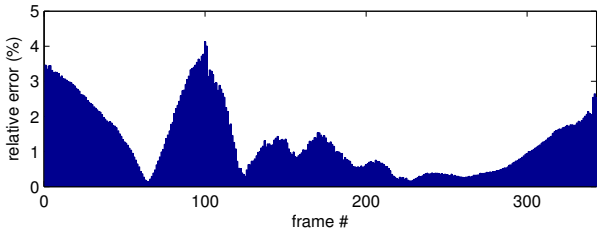

Fig. 13. Deviation of the trajectory from a perfect circle (least-squares fitted). Errors are relative to the radius of the circle.

TABLE I
TRACKING RESULTS: MEAN LOCALISATION ERROR (IN PIXELS) AND MEAN OVERLAP (IN PERCENTS). BOLD NUMBERS INDICATE THE BEST RESULT, UNDERLINED NUMBERS THE SECOND BEST.

	LGT	TLD	FoT	T	TMIC	TMAGIC
DOG	19.3/20	12.3/56	**4.7**/63	13.9/65	9.3/**71**	12.1/46
FISH	15.6/20	8.8/72	9.2/**75**	8.0/68	10.4/65	**5.6**/69
SYLVESTER	**13.1**/16	18.0/**58**	18.0/**58**	37.3/42	35.3/ 9	17.8/48
TWININGS	22.5/18	13.2/38	15.5/44	42.9/31	16.2/29	**9.1**/**53**
DOUGHNUT	131.0/35	**63.8**/56	181.7/44	544.3/12	814.1/ 7	87.9/**57**
RALLY-AUDI	**71.0**/37	18.0*/ 3	212.4/33	117.2/40	123.3/38	110.2/**48**
RALLY-LAN.	145.8/19	333.5/13	734.5/13	127.8/43	121.3/42	**94.3**/**53**
RALLY-OPEL	97.7/17	226.4/43	165.9/38	168.0/26	125.7/35	**44.0**/**52**
RALLY-OTS	**28.7**/22	89.8/25	116.2/29	72.0/34	102.4/27	48.9/**59**
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**
Average	61.4/26	92.1/39	154.4/42	114.0/42	138.7/36	**46.0**/**55**

*TLD on Rally-Audi reported loss of tracking after only 3 frames.

B. 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 most cases. Additionally, several new sequences of drifting cars were used, which have significant out-of-plane rotation (in addition to strong motion blur). These new sequences are available online including human annotated GT, evaluation scripts and a leader-board[3]. The initial frames are shown in Figures 15 and 16.

The TMAGIC tracker is compared with several state of the art tracking algorithms in Table I: LGT [4], TLD [14] and FoT [36]. The FoT tracker is similar to the proposed approach, in that it employs a group of independently tracked features with a higher management layer, however it operates in 2D only. GT in 3D is not available for these sequences: neither shape nor trajectory. Therefore the performance metric used was *localisation error*, i.e. the distance of the centre of the bounding box to the ground-truth centre, and additionally *overlap* of the (returned and ground-truth) bounding boxes. TLD can report an

[3]We invite anyone testing their tracker on these sequences to submit their results to this leader-board at http://cvssp.org/data/3DCars/.

Fig. 14. Example of failed reconstruction of the RALLY-VW tracking sequence by the CMP SfM WS [12].

Fig. 15. First frames from the test sequences (sequences from literature). From left: DOG [6], FISH [28], SYLVESTER [28] and TWININGS [3]. The initial bounding boxes are overlaid.

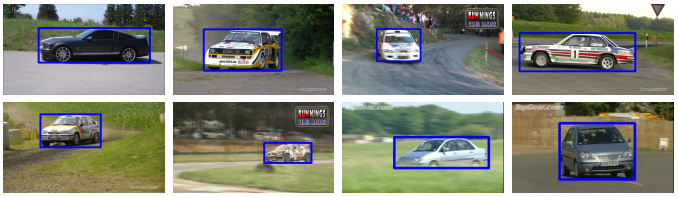

Fig. 16. First frames from the test sequences (newly created sequences). From top and left, in row-major order: DOUGHNUT, RALLY-AUDI, RALLY-LANCER, RALLY-OPEL, RALLY-OTS, RALLY-VW, TOPGEAR1 and TOPGEAR2. The initial bounding boxes are overlaid.

absence of the object. There are, however, no full occlusions in the tested scenes, thus in such cases the frames were assigned the maximal error found in the sequence and zero overlap. The results are visualised in Figure 17 and the mean values tabulated in Table I.

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. 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 lower, rendering it the best tracker for this sequence. The TWININGS sequence contains full rotation and was originally created to measure tracker robustness to out-of-plane rotation [3]. Unsurprisingly, TMAGIC significantly outperforms the state of the art on this sequence. The average localisation error reduction for all these scenes is 22 %.

The car 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 17). 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 to intelligently detect new

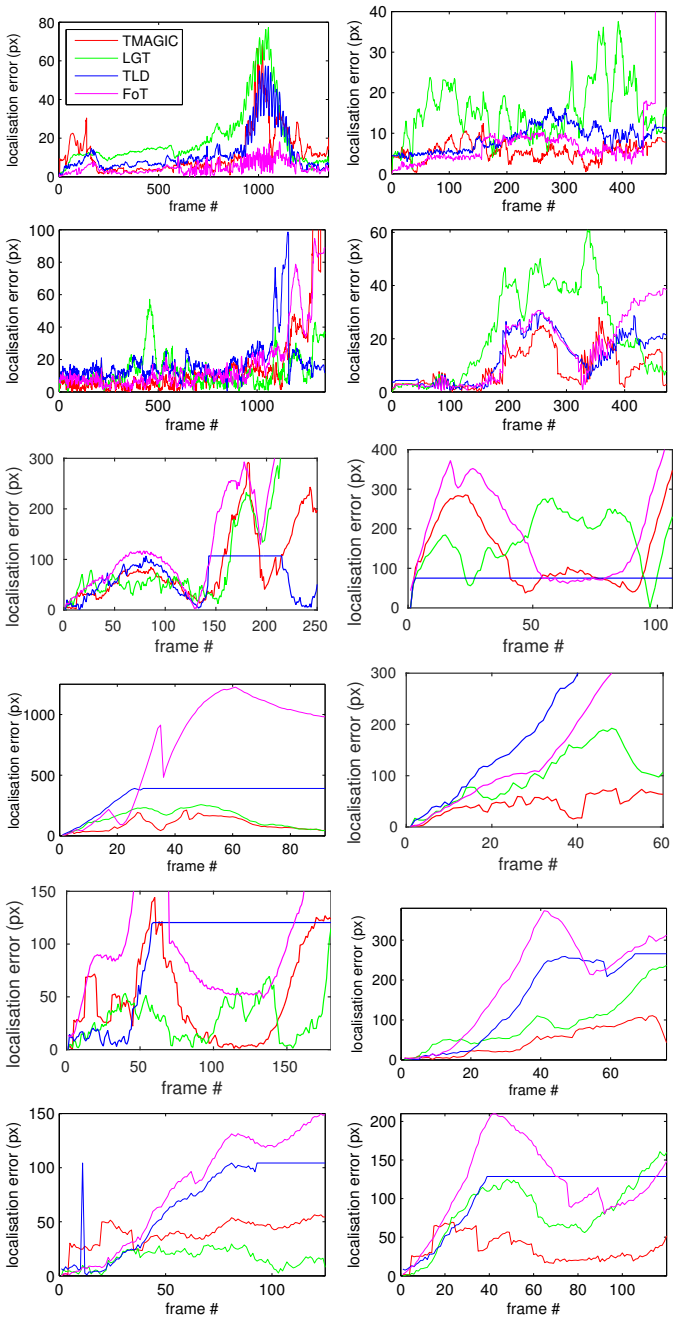

Fig. 17. Visualisation of results of the quantitative performance analysis. From top and left, in row-major order: DOG, FISH, SYLVESTER, TWININGS, DOUGHNUT, RALLY-AUDI, RALLY-LANCER, RALLY-OPEL, RALLY-OTS, RALLY-VW, TOPGEAR1 and TOPGEAR2.

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 46 % on average. Notice that the errors in the car sequences are generally higher, due to the higher resolution. The resulting model for the RALLY-VW sequence is visualised in Figures 1 and 5. 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. I show the effect of the different stages of TMAGIC on performance. Firstly, FoT is

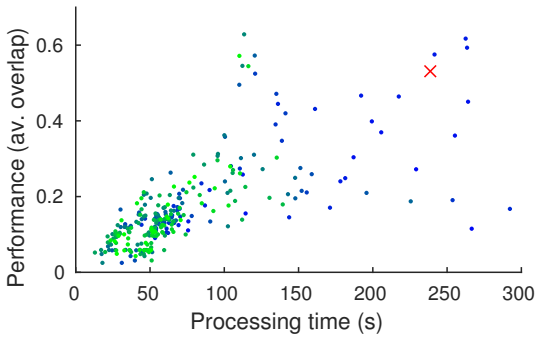

Fig. 18. Dependency of speed and performance on period of BA execution. Each data point is a result of a complete run on the RALLY-LANCER sequence, with the BA executed at fixed periods between 1 (blue) and 30 (green) frames. Results of the proposed execution strategy is shown as a cross.

compared with T, which assumes a fixed 3D model (sphere) and feature locations. This can be seen as 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, using the full system). This yields the most significant improvement (three-fold error reduction), 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.

C. Speed vs. Performance

As mentioned in Section V-A, Bundle Adjustment (BA) is executed on "keyframes" chosen according to the camera motion. There is a natural trade-off between processing speed and performance of the algorithm. Figure 18 illustrates this dependency on the RALLY-LANCER sequence. In this experiment, we executed BA periodically at specific frames. While executing it very sparsely can lead to speeds as high as 3 FPS, the performance is very low as the tracker easily loses the object without a realistic model. On the other hand, slower runs with more frequent BA have in general better performance. The correlation between execution time and average overlap is 0.49. For comparison, we show the result of the proposed on-demand execution as well, which outperforms any fixed interval BA approach.

D. Handling Concavities

One of the assumptions of the proposed algorithm is that the target is compact. Although convexity is *not* required, deep concavities may render the object outside of the star-shape class and/or make it difficult to find a suitable centre. In this section, resilience of TMAGIC against concavities is tested. A series of synthetic objects were created, which are of approximately spherical shape, with increasingly deep concavity in one side. Similarly to Section VI-A, this object rotates in front of the camera at a constant speed. Selected input frames and the extent of the concavity are visualised in Figure 19 together with the obtained results.

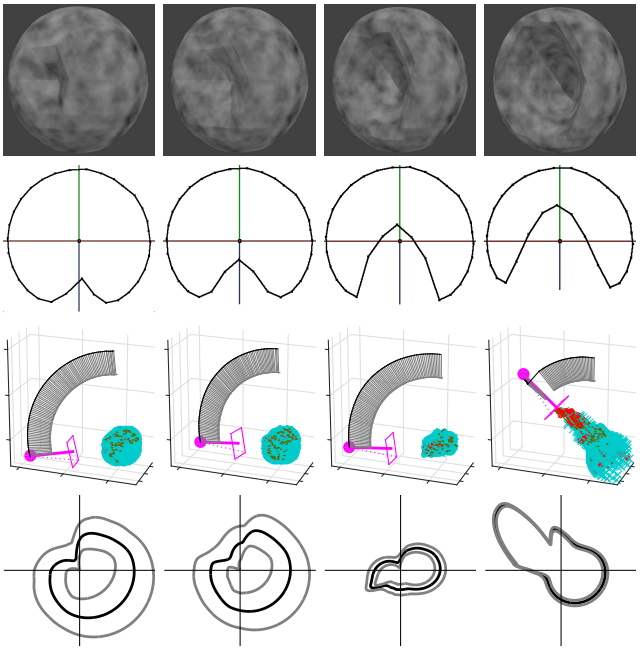

Fig. 19. Resilience of TMAGIC to concave targets. From top: Selected input frame, profile around the "equator", recovered camera trajectory and shape, recovered profile (mean surface ± 1 st. deviation).

In the first column, the concavity is not very deep, and TMAGIC tracks this sequence without difficulties, as can be observed from the realistic obtained model and near-perfect camera trajectory. Similarly in the second video (the second column), although the concavity reaches close to the centre of the target the performance is still good. In the third video, the concavity reaches even deeper, such that the object centre is outside its actual body. In this case the resulting model does not (even approximately) represent the shape of the target globally, however it is still useful for tracking – the camera trajectory is viable and from the 2D point of view, tracking is successful. Finally, the concavity in the last video (as shown in the fourth column) is so extensive that the object is a mere hollow "shell". TMAGIC starts tracking correctly, as can be seen near the top of the camera trajectory. However, it is unable to model the target shape once the concavity is aligned with the camera, and it eventually fails.

VII. CLOSING REMARKS

The experiments show that the TMAGIC 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. It is robust to small shape variations (e.g. a face), but is not capable of tracking articulated objects, e.g. a walking person. Another limitation would be full occlusions in long-term tracking. 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. TMAGIC assumes fixed camera calibration during the tracking. Cases of zooming in the sequences or cropped sequences usually do not cause tracking failures, but the resulting model is distorted. Online estimation of calibration parameters is a part of our ongoing related research.

TMAGIC tracks a single object. A naïve 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.

The sequences used for experimental validation in this work are publicly available, as well as the ground truth and an evaluation script. Anyone testing their tracker on these sequences is invited to submit their results to the online leaderboard.

REFERENCES

[1] GPc library. https://github.com/SheffieldML/GPc.

[2] S. Agarwal, K. Mierle, et al. Ceres solver. http://code.google.com/p/ceres-solver/.

[3] B. Babenko, M.-H. Yang, and S. Belongie. Robust object tracking with online multiple instance learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011.

[4] L. Cehovin, M. Kristan, and A. Leonardis. Robust visual tracking using an adaptive coupled-layer visual model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013.

[5] K. Chen, Y.-K. Lai, and S.-M. Hu. 3D indoor scene modeling from RGB-D data: a survey. *Computational Visual Media*, 2015.

[6] M. Chen, S. Pang, T. Cham, and A. Goh. Visual tracking with generative template model based on Riemannian manifold of covariances. In *Proceedings of the International Conference on Information Fusion*, 2011.

[7] O. Chum, J. Matas, and J. Kittler. Locally optimized RANSAC. In *Proceedings of the DAGM Symposium*, 2003.

[8] A. Dame, V. Prisacariu, C. Ren, and I. Reid. Dense reconstruction using 3D object shape priors. In *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[9] Y. Feng, Y. Wu, and L. Fan. On-line object reconstruction and tracking for 3D interaction. In *International Conference on Multimedia and Expo*, 2012.

[10] S. Hare, A. Saffari, and P.H.S. Torr. Struck: Structured output tracking with kernels. In *Proceedings of the International Conference on Computer Vision*, 2011.

[11] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.

[12] J. Heller, M. Havlena, M. Jancosek, A. Torii, and T. Pajdla. 3D reconstruction from photographs by CMP SfM web service. *Machine Vision and Applications*, 2015. http://ptak.felk.cvut.cz/sfmservice/.

[13] C. Hoppe, M. Klopschitz, M. Rumpler, A. Wendel, S. Kluckner, H. Bischof, and G. Reitmayr. Online feedback for structure-from-motion image acquisition. In *Proceedings of the British Machine Vision Conference*, 2012.

[14] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-Learning-Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2012.

[15] K. Kim, V. Lepetit, and W. Woo. Keyframe-based modeling and tracking of multiple 3D objects. In *Proceedings of the International Symposium on Mixed and Augmented Reality*, 2010.

[16] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Proc. of ISMAR*, 2007.

[17] A. Kundu, K. Krishna, and C. Jawahar. Realtime multibody visual SLAM with a smoothly moving monocular camera. In *Proceedings of the International Conference on Computer Vision*, 2011.

[18] K. Lebeda, S. Hadfield, and R. Bowden. 2D or not 2D: Bridging the gap between tracking and structure from motion. In *Proceedings of the Asian Conference on Computer Vision*, 2014.

[19] K. Lebeda, S. Hadfield, J. Matas, and R. Bowden. Texture-independent long-term tracking using virtual corners. *IEEE Transactions on Image Processing*, 2016.

[20] K. Lebeda, J. Matas, and O. Chum. Fixing the locally optimized RANSAC. In *Proceedings of the British Machine Vision Conference*, 2012.

[21] A. Mulloni, M. Ramachandran, G. Reitmayr, D. Wagner, R. Grasset, and S. Diaz. User friendly SLAM initialization. In *Proceedings of the International Symposium on Mixed and Augmented Reality*, 2013.

[22] R. Newcombe, D. Fox, and S. Seitz. DynamicFusion: Reconstruction and tracking of non-rigid scenes in real-time. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[23] R. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *Proceedings of the International Symposium on Mixed and Augmented Reality*, 2011.

[24] Q. Pan, G. Reitmayr, and T. Drummond. ProFORMA: Probabilistic feature-based on-line rapid model acquisition. In *Proceedings of the British Machine Vision Conference*, 2009.

[25] M. Pollefeys, L. van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch. Visual modeling with a hand-held camera. *International Journal of Computer Vision*, 2004.

[26] V. Prisacariu, O. Kahler, D. Murray, and I. Reid. Simultaneous 3D tracking and reconstruction on a mobile phone. In *Proceedings of the International Symposium on Mixed and Augmented Reality*, 2013.

[27] C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2004.

[28] D. Ross, J. Lim, R.-S. Lin, and M.-H. Yang. Incremental learning for robust visual tracking. *International Journal of Computer Vision*, 2008.

[29] L. Sigal, M. Isard, H. Haussecker, and M. Black. Loose-limbed people: Estimating 3D human pose and motion using non-parametric belief propagation. *International Journal of Computer Vision*, 2012.

[30] P. Smith, I. Reid, and A. Davison. Real-time monocular SLAM with straight lines. In *Proceedings of the British Machine Vision Conference*, 2006.

[31] N. Snavely, S. Seitz, and R. Szeliski. Photo tourism: Exploring image collections in 3D. In *Proceedings of the ACM Special Interest Group on Computer Graphics and Interactive Techniques*, 2006.

[32] N. Snavely, S. Seitz, and R. Szeliski. Modeling the world from internet photo collections. *International Journal of Computer Vision*, 2007.

[33] S. Song and J. Xiao. Tracking revisited using RGBD camera: Unified benchmark and baselines. In *Proceedings of the International Conference on Computer Vision*, 2013.

[34] P. Torr and A. Zisserman. MLESAC: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 2000.

[35] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. http://www.vlfeat.org/, 2008.

[36] T. Vojíř and J. Matas. The enhanced flock of trackers. In *Registration and Recognition in Images and Videos*, Studies in Computational Intelligence. 2014.

[37] R. von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall. LSD: A fast line segment detector with a false detection control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2010.

[38] C. Wojek, S. Walk, S. Roth, K. Schindler, and B. Schiele. Monocular visual scene understanding: Understanding multi-object traffic scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013.

[39] C. Wu. Towards linear-time incremental structure from motion. In *Proceedings of the International Conference on 3D Vision*, 2013.

[40] C. Wu, S. Agarwal, B. Curless, and S. Seitz. Multicore bundle adjustment. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[41] Z. Yin and R. Collins. On-the-fly object modeling while tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[42] L. Zhang. *Line Primitives and Their Applications in Geometric Computer Vision*. PhD thesis, Department of Computer Science, Kiel University, 2013.