Dense Rigid Reconstruction from Unstructured Discontinuous Video

Karel Lebeda Simon Hadfield Richard Bowden
University of Surrey, Guildford, GU2 7XH, United Kingdom
{K.Lebeda,S.Hadfield,R.Bowden}@Surrey.ac.uk

Abstract

Although 3D reconstruction from a monocular video has been an active area of research for a long time, and the resulting models offer great realism and accuracy, strong conditions must be typically met when capturing the video to make this possible. This prevents general reconstruction of moving objects in dynamic, uncontrolled scenes. In this paper, we address this issue. We present a novel algorithm for modelling 3D shapes from unstructured, unconstrained discontinuous footage. The technique is robust against distractors in the scene, background clutter and even shot cuts. We show reconstructed models of objects, which could not be modelled by conventional Structure from Motion methods without additional input. Finally, we present results of our reconstruction in the presence of shot cuts, showing the strength of our technique at modelling from existing footage.

1. Introduction

In this paper we address the task of reconstructing a 3D model of an a priori unknown object from unconstrained, unstructured and discontinuous video such as broadcast footage. With such generic input, this is a challenging task, since the target usually occupies only a small portion of the video frame and presents issues with both low resolution of the object and challenges with the background (distractors in the scene and other, possibly similar, objects). This problem has been extensively studied by the visual tracking community. We exploit recent advances in tracking literature to handle this issue. Our dual-model approach allows us to coarsely model the object shape online for target/background segmentation (following the tracking paradigm) and at the same time provide a high-quality model (reconstruction) with super-resolution texture as the output for subsequent applications.

The last decade has brought a previously unimaginable boom in the applications of custom 3D models. To name just a few, we confine ourselves to the examples of video games and 3D printing. Recently, video games have appeared offering scanned, reality-based locations and assets instead of manually modelled ones (such as ReRoll [2] or The Vanishing of Ethan Carter [3]). Thus far, the creation of the models has largely been in the hands of game developers, however this can be expected to change in the near future. In the field of 3D printing, open-source, partially self-replicating systems such as RepRap [15] have emerged, making 3D printing of scanned models more accessible.

Figure 1. The HILLCLIMB sequence, a broadcast video divided into many sub-sequences by shot cuts. We provide an automatic modelling algorithm working through these sub-sequences.

For such consumer-grade applications, there are different scanning requirements than for a professional system. High precision and realism are less important than accessibility without specialised equipment, such as laser scanners or camera rigs. Although software solutions exist to allow the home user to create 3D models from consumer cameras (VisualSFM, 123D Catch or Python Photogrammetry ToolBox, to name a few), they have numerous limitations. The scene needs to be strongly constrained: perfectly static, with constant illumination, *etc*., and the object of interest must be the dominant feature of the scene. While these requirements can be met in cases of hand-held capture, they are unrealistic to expect from existing footage or casual everyday settings. An example might be the reconstruction of objects from broadcast footage, archive footage or online-sourced videos.

As any camera is a line-of-sight device, a common artefact in reconstruction are holes in the model where observations are absent and therefore no reconstruction is possible. For hand-held capture the user can rescan these areas to get

a watertight (*i.e*. closed surface with no holes) model without the need to start the capture from scratch. However, this is not possible for pre-recorded or broadcast footage; as an alternative multiple shots could be combined, leading to possibilities such as post-capture fusion and/or guided model refinement. Figure 1 shows a broadcast video of a rally race, where one car is followed through 18 shots (subsequences). Furthermore, the car is very rarely the major element of the scene as it typically covers only between 1 and 10 % of the frame. We show how our dual-model approach allows for successful modelling of this unstructured, unconstrained, discontinuous video-sequence.

Our contributions are as follows. Firstly, we show a novel technique for dense modelling of moving objects (identified by a rough bounding box in the first frame) in an unconstrained, unstructured scene, which actively avoids background distraction by employing a dual-model approach (Sections 4 and 5). Secondly, we show how it is possible to combine evidence from different parts of discontinuous video, effectively overcoming shot cuts (Section 6). Finally, we demonstrate the abilities of our technique on a challenging video-sequence (Section 7) and make both the sequence and the resulting model publicly available (including partial models leading to the final output) [1].

2. Related Work

The main difference between our approach and conventional Structure-from-Motion (SfM) techniques is firstly its online nature and secondly the active background segmentation. While computing the overall scene geometry, SfM extracts the dominant motion and discards outliers. However, as we mentioned earlier, the object of interest can easily be as small as 1 % of the frame area. In such a case it is completely ignored by SfM algorithms and the background scene is reconstructed instead. Our algorithm, however, segments the object from the background, avoiding such distraction. The same problem occurs in many recent methods for monocular visual SLAM (Simultaneous Localisation And Mapping). SLAM relies on a static scene and reconstructs its geometry and camera motion, while our objective is to reconstruct a 3D object as it moves between multiple uncalibrated cameras. Furthermore, SLAM systems usually do not provide dense reconstruction of the scene. See the rest of this section for particular relevant techniques in recent literature.

As mentioned previously, one of the main differences between our technique and conventional SfM is its online nature. There have been several exploratory works investigating online SfM, mostly aiming to provide feedback to a human operator, performing the scanning procedure. While ProFORMA by Pan *et al*. [26] builds a partial coarse 3D model by tetrahedration of a cloud of sparse features to reveal unseen parts of a scanned object/scene, the work of Hoppe *et al*. [13] goes further and provides the operator with a redundancy map of the reconstructed area. There have been numerous works using depth sensors [18, 19, 25] and also commercial scanning software such as Skanect [31], however these are out of scope of this paper, as they cannot be applied to archive/broadcast data and they limit portability.

Another related work is Dense Tracking And Modelling (DTAM [24]), standing between SfM and SLAM. It offers a real-time reconstruction of a static scene based on a monocular video stream, using a variational approach. As previously mentioned, all of these assume the modelled object covers the majority of the scene and the distractors can be avoided by a simple outlier rejection. As explained in the previous section, this is not the case for a large portion of video footage, which could be used for 3D reconstruction.

Modern approaches to monocular visual SLAM [23, 30] try to remove this assumption. However, while increasingly focusing on dynamic scenes, the main aim of SLAM is increased robustness against occlusions. Although recent approaches such as ORB-SLAM [23] offer excellent performance in cases of strong occlusion, the requirement that the modelled target is the only part of the scene being preserved in time is restrictive.

In the area of visual object tracking, 3D based approaches offer extraction of 3D shape and trajectory even of objects represented only by a minority of the video frame. Kundu *et al*. [20] use motion segmentation to track and reconstruct moving bodies in their SLAM pipeline. Feng *et al*. [8] introduced the idea of 3D monocular tracking with no offline modelling or training, using a colour-based segmentation. To avoid the assumption of a distinctive colour of the target object, TMAGIC by Lebeda *et al*. [21] used machine learning techniques to model the 3D surface shape, providing the segmentation. This also removes the need for initialisation, making a bounding box in the first frame the only user input. Our approach is partially inspired by these, however two major differences exist. Instead of keeping a cloud of features for the purpose of tracking, we aim to extract as much spatial information as possible, to obtain detailed models of the reconstructed object. Furthermore, processing discontinuous sequences breaks the assumption of small inter-frame motion, required by these techniques.

3. Algorithm Overview

The requirements of a 3D modelling algorithm are relatively straightforward – maximal realism in reconstructed shape and texture. To distinguish between the foreground and the background, we build a partial model online during processing of the video-sequence. This model, known as the *object model* in tracking, has however a completely different set of priorities. While we need only a coarse approximation of the shape, it is beneficial to have a confi-

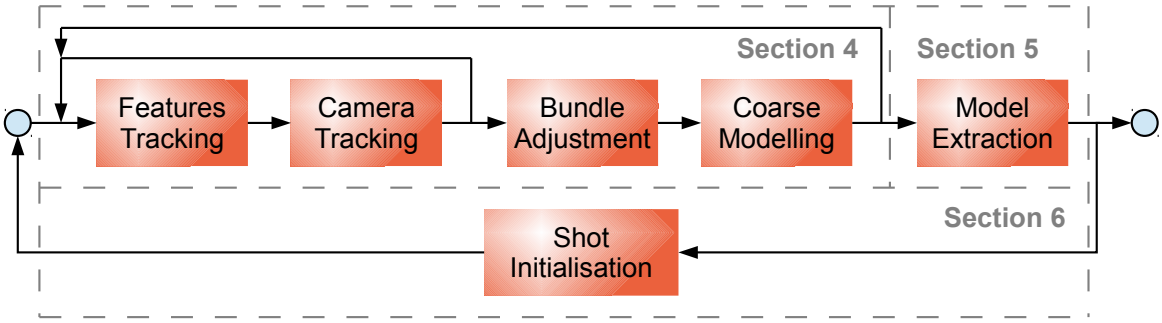

Figure 2. Flow of the proposed algorithm. See section 3 for description.

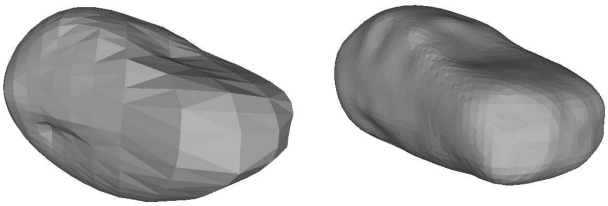

Figure 3. Sparse and dense model in our dual-model approach.

dence measure in every point of the model. Furthermore, we need this model to smoothly extrapolate across unseen areas, to create a compact, watertight surface at all times. For this reason we employ a dual-model approach, building a coarse, probabilistic model online during processing and a fine, topological one, as an output from each (sub)sequence. On the coarse level, we model the object shape as a Gaussian Process (GP) in spherical coordinates. For the final topological model, we have chosen a traditional triangular mesh. See Figure 3 for a comparison of the two models. The coarse model has much lower level of detail, no texture and no explicit discrete vertices (the vertices for visualisation were sampled regularly in the angular space).

As visualised in Figure 2, the proposed algorithm consists of three nested loops. The first, tracking loop, is performed on a frame-to-frame basis and consist of tracking of sparse and dense feature clouds and of estimation of the camera pose based on these. The second, reconstruction loop, is performed only after some camera motion is observed to achieve "keyframes" with equidistant camera poses in the 3D space. This loop includes global joint optimisation of 3D feature clouds and camera poses (Bundle Adjustment, BA), and an update of the coarse model. In cases of continuous videos, this is followed by an extraction of the final model and its texture. In cases where the video consists of several discontinuous shots, the model is aligned with the first frame of the new sub-sequence and used to initialise processing of this sequence in the outermost, re-initialisation loop. As the scene may contain many different objects, the one of interest is marked by the user by a bounding box in the first frame.

4. Online Dense Modelling

In this section, we explain the two inner loops. Firstly the tracking loop with the features used and camera trajectory estimation and secondly the reconstruction loop, where the bundle adjustment is executed and the coarse model is trained.

4.1. Tracking Keypoints and Camera

The standard approach in 3D reconstruction is to compute the scene geometry based on sparse feature correspondences and then triangulate a dense cloud as a post-processing step. Alternatively, sparse features can be densified by iterating *expand and filter* steps [10]. However, in our approach we aim for online processing, returning to earlier frames only for the purpose of texturing the obtained model (see Section 5). Therefore, we take inspiration from non-rigid SfM approaches (such as [9, 11]), which often uses *dense trajectories* as a basic building block. Therefore, in addition to long-living sparse features (keypoints), robust to drift and providing long-term global geometric consistency (including *loop closures*), we also use dense trajectories with a very limited life span. These features provide mainly input data for modelling and they offer better coverage and higher level of details in the reconstruction. The idea of integrating multiple feature types has been explored *e.g.* in [21, 29].

Figure 4 illustrates the life cycle of sparse features, with transitions S1–S4. These are extracted from the image as SIFT features (S1) and tracked from frame to frame using the pyramidal Lucas-Kanade tracker. The currently estimated coarse model is used for object/background segmentation to filter the features. When the LK tracker cannot converge, the feature is assumed to be invisible (*e.g.* occluded, S2). The same transition from the *active* to the *invisible* state happens when self-occlusion is detected. This is indicated by local surface normals pointing away from the camera when the currently estimated object model lies between the camera and the features in the 3D space.

Similarly, if the surface normal indicates that an *invisible* feature lies on a side of the object which is currently

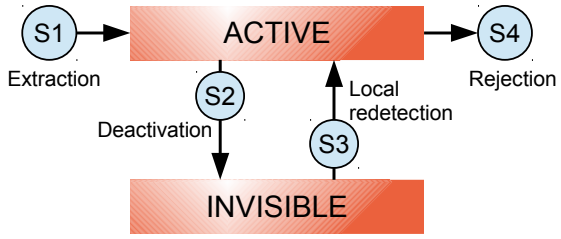

Figure 4. Life cycle of sparse features.

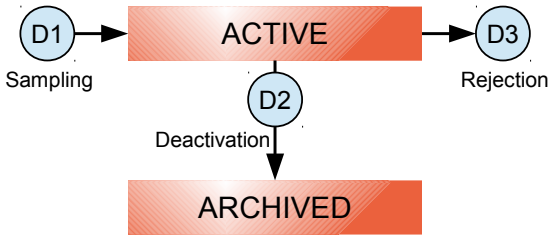

Figure 5. Life cycle of dense features.

visible, a *local redetection* (S3) is attempted. An image patch (extracted and stored from the frame when the keypoint is created) is warped to account for viewpoint change and searched for in the current frame. This is done in the neighbourhood of the expected (projected) position and size again by Lucas-Kanade tracking (hence *local*; we discuss *global* redetection in Section 6). There are also occasions, when a feature can be *invalidated* and thus stop being used (S4). This is when it becomes inconsistent with the remainder of the cloud in either its 2D motion or 3D location. The 2D motion is verified by the epipolar geometry computed from all *active* features. The 3D location is verified by the global 3D shape model. This is vital for the algorithm, as it allows the rejection of background features without limiting adaptation of the model.

Figure 5 illustrates the life cycle of dense features, with transitions D1–D3. These are sampled from the image on a regular grid (D1). The grid size is based on the image resolution and required density of the model. There is a trade-off between resulting quality and processing time, so the sampling density can be used as a user-defined parameter. The dense trajectories are estimated on a frame-to-frame basis from a CNN based dense optical flow (FastDeepFlow [32]). This is prone to drift, thus we keep the dense trajectories short (20 frames were used in out experiments). For this reason there is no *invisible* state. Dense features are either successfully tracked during their life span, after which they are transferred (D2) into the *archived* state or discarded as outliers (epipolar constraint, D3). Features in the *archived* state are used for 3D modelling, but not tracked any more.

The 3D trajectory and shape is reconstructed as follows. From the 2D trajectories and respective 3D positions of the features, the 3D trajectory of the camera is recovered using an optimisation approach (conditional gradient method), minimising the sum of robust squares of projection error. In frame t, the camera C^t is obtained as

$$\mathrm{C}^t = \arg\min \sum_{q\in\{S,D\}} \sum_{\mathbf{X}_i\in\mathcal{X}_q} w_q \rho\left(||\mathrm{C}^t(\mathbf{X}_i) - \mathbf{x}_i^t||^2\right) , \tag{1}$$

where ρ is a robust cost function and the flag q distinguishes between sparse (S) and dense (D) features to choose the appropriate weight (see below). Feature $\mathbf{x}_i^t$ has in the 3D feature cloud $\mathcal{X}_q = \{\mathbf{X}_i\}$ its correspondence $\mathbf{X}_i$, which is projected by the t-th camera as $\mathrm{C}^t(\mathbf{X}_i)$. In order to achieve online modelling, the whole system is regularly optimised via bundle adjustment (BA), using solver [4]. If we denote the set of all cameras as $\mathcal{C} = \{\mathrm{C}^t\}$, it can be formalised as:

$$\mathcal{C},\mathcal{X} = \arg\min \sum_{\mathrm{C}^t\in\mathcal{C}} \sum_{q\in\{S,D\}} \sum_{\mathbf{X}_i\in\mathcal{X}_q} w_q \rho\big(||\mathrm{C}^t(\mathbf{X}_i) - \mathbf{x}_i^t||^2\big), \tag{2}$$

The bundle adjustment is executed on keyframes of the trajectory, equidistant in terms of 3D camera location.

There is a significant imbalance between the numbers and importance of sparse and dense features. For this reason, one might want to give them different weights (w_S and w_D) during the process – in camera tracking, bundle adjustment, model creation, *etc*. In all these cases, we have downweighed the dense features by a factor 0.01 in our experiments. This reflects their numbers (usually 1–2 orders of magnitude more than that of sparse features) and the notion of dense features being omnipresent but less reliable.

4.2. Coarse Online Modelling

For online target/background segmentation, we train a Gaussian Process (GP) as our coarse, probabilistic model. The representation was chosen such that every point on the surface is represented in spherical coordinates (θ, φ) relative to the object centre γ:

$$\mathbf{X} = \gamma + r \cdot (\sin\theta\,\cos\varphi, \sin\theta\,\sin\varphi, \cos\theta)^\top \tag{3}$$

(for more details on the choice of γ see below). For any pair of angles, the radius is modelled:

$$r = \mathrm{GP}(\theta, \varphi|\kappa) , \tag{4}$$

where κ is the *kernel* of the GP. This means 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 we query the model at regularly sampled positions (see Figure 3, left). This, however, is not an obstacle for its use in our framework. As training data, the 3D features (sparse in both *active* and *invisible* states and dense in *archived* state) are used.

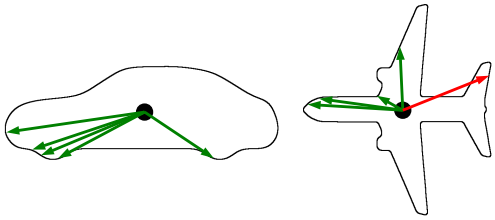

Figure 6. Star domain example in 2D. Left: Every region of the car shape can be reached from the *centre* without crossing the boundary, *i.e*. its shape is a star domain. Right: Since there are regions unreachable by a straight line, it is not a star domain.

As mentioned previously, the Gaussian Process shape model is fully probabilistic. That means that we have not only a shape estimate, but a whole distribution of shapes (radius functions). From this distribution we use the mean (*i.e*. the most probable) shape as the estimate and the variance as the uncertainty at any given point of the object surface. Other beneficial properties of Gaussian Process modelling are high robustness to overfitting, smooth interpolation and extrapolation in regions without training inputs. Its non-parametric nature also makes it possible to model wide range of object shapes and resolutions without the need for reparameterisation. To specify rigorously what class of objects we are able to model, one needs to consider the properties of the spherical representation. Since the radius for any given set of direction angles 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 the final (polygonal) model when the online model smooths over minor regions which break this assumption.

Attention must be paid to the selection of the *shape centre* γ. 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 we employ a data-driven shape centre as follows. We use the centre of mass of the training points only as an initialisation and then shift the centre towards the midpoint between this and the centre of mass of the sampled points (trained with the previous centre):

$$\gamma_{\text{new}} = \alpha \frac{\overline{\mathbf{X}} + \overline{\mathbf{M}}}{2} + (1 - \alpha)\gamma \,, \tag{5}$$

where α is a learning factor (set to 0.5 in our experiments)

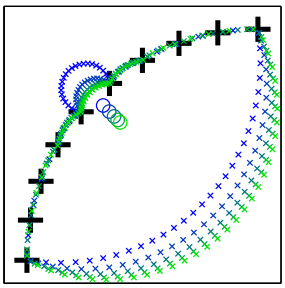

Figure 7. Iterative search for the shape centre. Black: training data. Coloured: shape centre ($\circ$) and sampled points ($\times$), iterating from the centre of mass $\overline{\mathbf{X}}$ (blue) to convergence (green).

and $\mathbf{M}_i$ is a point sampled on the surface model (visualised as $\times$). 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, we perform one step 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 our prior knowledge about properties of the modelled function (in our case surface shape), especially smoothness and differentiability. One of the most commonly used is the RBF (Radial Basis Function, also called Gaussian) kernel, which is infinitely differentiable and therefore induces smooth shape contours. This, however, causes artefacts in the shape, as the overly smooth gradient exaggerates rapid radius changes. The same holds for the Matérn kernel (of both common orders 3/2 and 5/2). For this reason, we employ the *exponential* kernel, 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:

$$\kappa = \kappa_{\text{Exp}} + \kappa_{\text{B}} + \kappa_{\text{W}} \,. \tag{6}$$

Besides this choice of kernel, there are no other parameters in the GP modelling.

As mentioned, we retrain the coarse model every time the feature cloud changes, *i.e*. after every bundle adjustment. This is then used in several different ways. The model segments the video frame into the region occupied by the target and the background. To test if an image location lies inside the occupied region, we back-project this location's coordinates as a ray in the 3D space and intersect it with the model. If such an intersection exists, the pixel location would be inside of the projected image of the shape model and we assume the point (*e.g*. a newly generated feature)

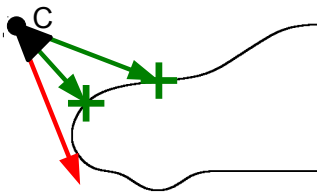

Figure 8. Model intersection search example. The intersection points (+) are initial 3D locations of the newly generated features. C marks the camera centre.

is a part of the object. When the ray does not intersect the model, it is assumed to belong to a background location in the image.

Besides the object/background segmentation, we use the intersections as an initialisation of the 3D locations of newly generated features. These locations are later refined in bundle adjustment. See Figure 8 for an intuitive example. Additionally, the model gives us an estimate of surface normals, which is necessary for later reconstruction of the final model. This is done by sampling several points in close proximity to the location of interest (*i.e.* several orders of magnitude below the object dimensions) and locally fitting a tangent plane. Finally, this model, which is learned online as the video-sequence is processed, can be shown to the user as immediate feedback (analogous to [13, 26] with handheld or remote-controlled aerial recording) on how successful the modelling has been thus far.

5. Model Extraction

The final model is the output of our approach, together with the object trajectory (both in 2D and 3D). The model needs to be *explicit*, *i.e.* consisting of particular samples (vertices) and their relationships (edges). As with the coarse model, we use sparse features in both *active* and *invisible* states and dense features in the *archived* state for modelling. When modelling an object from a continuous video, this is the final stage of our technique. However, when a shot cut is reached (which may be detected automatically [6]) while processing the input sequence, we need to re-detect the object, as part of the outer loop of our algorithm. This is where the final, polygonal model extracted from the first sequence, is used.

There are numerous approaches to reconstruct a surface model from a set of scattered points, such as marching cubes [22], ball pivoting [5] or methods based on Moving Least Squares [28]. For an extensive study we recommend [33], evaluating a broad range of techniques and summarising their properties. In this work we use the Poisson reconstruction [16, 17], in its screened variant. Input points and their normals are interpreted as samples of the *indicator function* χ (resolving the inside/outside problem) and a vector function $\vec{\mathbf{v}}$, respectively. The implicit indicator function χ is found as a solution of the Poisson equation, such that its gradient $\nabla\chi$ approximates the sampled normals $\vec{\mathbf{v}}$:

$$\Delta\chi = \nabla\cdot\nabla\chi \;\approx\; \nabla\cdot\vec{\mathbf{v}}\,. \tag{7}$$

Poisson reconstruction provides a global solution to this approximation and hence to the reconstruction problem and therefore smoothly fills even large gaps in the surface. This is the main reason why it is used in this work, as we require a watertight surface. The set of all the points on the model (χ_0-isosurface) is referred to as

$$\mathcal{M} = \{\mathbf{X}|\chi(\mathbf{X}) = \chi_0\}\;, \tag{8}$$

where χ_0 is chosen as the mean χ of the training points.

Once the shape model has been obtained from the 3D feature clouds, its *texture* is extracted. To achieve this we need to process the input sequence once more, project the final model into every frame and interpolate the colour information from the video. For each pixel of the texture we compute the colour as the median of the observations from all frames (cameras $\mathcal{V}(\mathbf{X})$) where the particular polygon was visible:

$$\mathtt{T}(\mathbf{X}) = \underset{\mathrm{C}^t\in\mathcal{V}(\mathbf{X})}{\mathrm{Median}}\left(\mathtt{I}^t(\mathrm{C}^t(\mathbf{X}))\right) \quad \forall\mathbf{X}\in\mathcal{M}\,, \tag{9}$$

$$\mathcal{V}(\mathbf{X}) = \{\mathrm{C}\in\mathcal{C}\,|v(\mathbf{X}|\mathrm{C})\}\;, \tag{10}$$

where $v(\mathbf{X}|\mathrm{C})$ is a function indicating the visibility of point $\mathbf{X}$ by camera C. To avoid extracting unreliable information, *e.g.* near the edges of the object, we require the angle between the surface normal and the ray from the camera centre to be below a given threshold (85° in our implementation). The texture is extracted at a resolution specified by the user. With precise camera tracking and a high-detail shape model, it is possible to sample points at resolutions exceeding the original video, leading ultimately to a 3D super-resolution model.

6. Modelling Across Video Discontinuities

Modelling thus far was limited to a single sequence where small motions can be tracked at the feature level via Lucas-Kanade. However, to build a compound model from multiple sequences or over shot cuts in broadcast video we need to combine information from discontinuous shots.

There are two basic approaches to this problem. Firstly, the sub-sequences (shots) can be each processed independently and the final models merged as a post-processing step. While this has the advantage of not needing the re-alignment of the partial model with every new subsequence, there is the challenge of aligning the partial meshes, which may have only a small overlap (as the parts of the object visible in one shot may be mostly unseen in others). Furthermore, the evidence in a single shot may not

be sufficient for even a partial reconstruction (*e.g.* due to insufficient camera motion). Finally, this approach needs a user-given initialisation for every sub-sequence. Another possible approach is to re-detect the object at the beginning of a new sub-sequence and continue processing from that point, using the information gathered from the previous parts of the footage. This alleviates problems with insufficient information present in any particular shot (except the very first one) and no matching and alignment of partial models is needed. However, the problem of aligning the model gathered thus far with the new sub-sequence needs to be resolved.

Since our aim is to provide an automated approach with minimal user input, we employ the latter. Having a good initial model also allows us to process sub-sequences which otherwise would be intractable due to occlusions or camera motion with too small baseline across the shot.

To achieve this, we first obtain a rough alignment of the model with the first frame of the next sub-sequence (*i.e.* the camera pose in that frame considering a fixed model), using our sparse feature cloud and keypoints independently detected in the frame. On these 2D-to-3D matches, P3P-RANSAC is executed. The inliers (consistent matches) to the camera pose are taken as *visible* sparse features after the reinitialisation.

Since we have the textured model, we can refine the pose using every model point and not just a sparse subset. This is done (using a conditional gradient method) as follows. We define the *brightness constancy* constraint as:

$$\mathtt{T}(\mathbf{X}) = \mathtt{I}^t\left(\mathrm{C}^t(\mathbf{X}|\beta^*)\right) \quad \forall \mathbf{X} \in \left\{\mathbf{X}_i \in \mathcal{M} \,\middle|\, v(\mathbf{X}_i|\mathrm{C}^t)\right\}, \tag{11}$$

i.e. for each visible 3D point $\mathbf{X}$, its colour in the texture $\mathtt{T}$ must be the same as the colour in the image $\mathtt{I}$ where $\mathbf{X}$ is projected using camera C parameterised by the true (unknown) β^*. We approximate the brightness constancy by a first-order Taylor expansion. This gives us a simple equation:

$$\mathtt{T}(\mathbf{X}) \approx \mathtt{I}^t\left(\mathrm{C}^t(\mathbf{X}|\beta)\right) + \nabla\mathtt{I}^t\mathtt{J}_\mathrm{C}\Delta\beta\,, \tag{12}$$

which is similar to the optical flow constraint [14], including the projection function and its Jacobian $\mathtt{J}_\mathrm{C}$. We solve via linear least squares for an unknown step in the camera parameters $\Delta\beta$. Since the formulation is in terms of intensity and we typically work with coloured (RGB) images, the total number of equations is 3 times the number of sampled pixels. We initialise this optimisation with the solution of the P3P-RANSAC and iterate until convergence, optimising the alignment of the textured model with the first frame of the new sub-sequence. A multi-scale approach is taken, solving on a blurred image in the early steps.

The new sequence is processed using this initialisation, without the need for human intervention. The first step after the global re-detection is a local re-detection, to maximise the number of visible existing-model features used and therefore the accuracy of camera pose estimation in the first frames. After it has been processed, the new feature cloud is integrated into the original one and a new, more detailed and complete polygonal model is created.

7. Results

To evaluate performance of our algorithm, we first show it in the short-term scenario, *i.e.* one continuous video where the target is fully visible in all the frames. See Figure 9 for example frames and our results on several video-sequences used in recent publications. Model regions, which were unseen in the original sequence and are therefore untextured, are marked by bright green colour.

For comparison, we show reconstructions made by the *CMP SfM WebService* [12] in Fig. 10. In the first column, the reconstruction from the raw video can be seen, with only a fraction of the car partially reconstructed. When supplied with GT bounding boxes at every frame to segment out the background and focal length estimates (provided in $\mathcal{C}$ from our BA), it yields the model shown in the second column. Our approach (last column) returns significantly cleaner results automatically without such extensive user interaction. We show the textured variant of our sparse GP model (roughly equivalent to [21]) in the third column. An example of an untextured resulting model is shown in Figure 3 and further in the supplementary video D (including the sparse models).

To test the ability of the proposed algorithm to model the geometry of long sequences including shot cuts, we show the modelling results on multiple sub-sequences of the HILLCLIMB sequence. As new shots are added to the reconstruction, the model becomes slowly more detailed and complete. See Figure 11 for the results. The first sequences

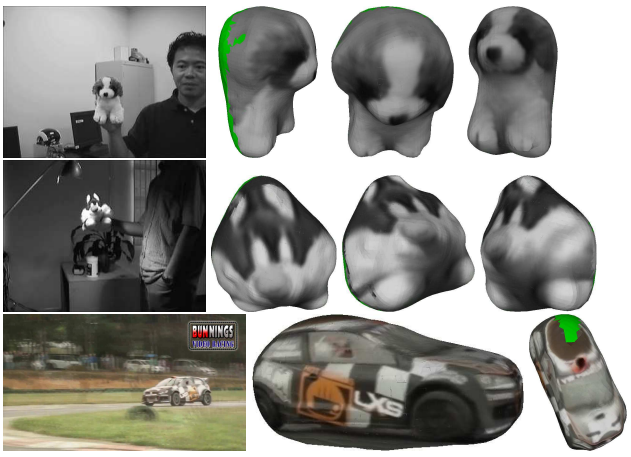

Figure 9. Resulting models on sequences from literature with varying resolution. From top to bottom: DOG1 (90 px [7]), SYLVESTER (50 px [27]) and RALLY-VW (410 px [21]).

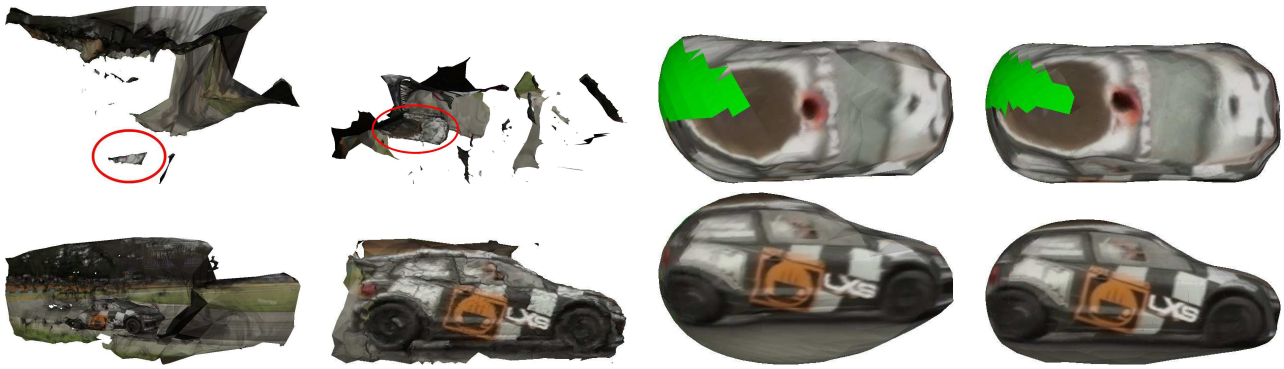

Figure 10. Results on the RALLY-VW sequence. Top and bottom rows show the top and side views of models from the same method (the bottom row is manually cropped for CMP results). Left to right: CMP SfM WebService [12] (raw video input); CMP SfM WebService [12] with added information (top: whole scene, bottom: manually cropped); textured sparse GP model; full final model (proposed).

(a) After 1 sequence. (b) After 2 sequences. (c) Final model.

Figure 11. Progressive growth of the model after processing particular shots from HILLCLIMB. The first two models and the final model are shown.

track the car mostly from the front, *i.e*. the rear parts are missing information. However, the later addition of subsequences covering the rear of the car incorporates these missing regions.

One region which remains unmodelled after all the sequences are processed is the bottom of the model, which is completely unseen. This is, however, an inherent property of this dataset and cannot be addressed without human intervention.

A natural way of processing the sequence is in temporal order, *i.e*. the first shot first and then the rest as they follow. However, changing the ordering can be beneficial. We used one such heuristic for the ordering of the sub-sequences. After breaking the original sequence into shots, we start with the longest one, as it can be expected to yield the most complete model. Subsequent sequences are then chosen in the order of decreasing quality of alignment, *i.e*. we run the P3P-RANSAC on all the sequences and choose the one with the highest number of inliers (successfully matched features).

In the supplementary material, additional results are available in the form of videos. Furthermore, the original data of the HILLCLIMB sequence will be published on the authors' website [1].

8. Conclusions

We have presented an algorithm for automated reconstruction of 3D models from unconstrained, unstructured, discontinuous videos. It provides a detailed, textured model of an a priori unknown object with the only user input being a single bounding-box initialisation of the object to be reconstructed – even from videos consisting of several shots. It actively avoids modelling the scene background, removing the assumption that the object of interest covers most of the frame.

The modelling approach, as presented, has several limitations. One of the major assumptions of this work is that the target object is rigid. Extension to an unconstrained non-rigid reconstruction is a planned subject of our future work. Resolution of the video-sequence is another limiting factor. While it is possible to obtain a 3D camera trajectory and a coarse model of objects at resolutions as low as 320×240 px (*e.g*. Sylvester in the eponymous sequence is approximately 50 px in size – see Figure 9), the resulting models are blob-like with low levels of detail; therefore higher resolution is recommended. Most of the examples shown in this paper come from videos with the resolution 1280×720 px.

Finally, the reconstruction is thus far limited to star domain shapes (technically only in the case of the sparse model, however the final model is influenced by this as well). This may be addressed in the future work, for instance using an intermediate reparameterisation layer.

In our future work, we would like to extend and generalise our approach in several different ways. As already mentioned, dense non-rigid reconstruction is one of our long-term goals. Current approaches to non-rigid SfM usually use videos taken under very constrained conditions (*e.g*. only one side of the object covered). Our aim is to allow reconstruction in any setting, with an unconstrained camera.

Acknowledgement: This work was supported by the EPSRC project EP/I011811/1, and the Rabin Ezra Fund.

References

[1] Authors' website: the HILLCLIMB sequence and additional results. http://cvssp.org/Personal/KarelLebeda/TMAGIC/dense/. 2, 8

[2] ReRoll game. http://rerollgame.com/. 1

[3] The Vanishing of Ethan Carter game. http://ethancartergame.com/. 1

[4] S. Agarwal, K. Mierle, et al. Ceres solver. http://ceres-solver.org. 4

[5] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. The ball-pivoting algorithm for surface reconstruction. *VCG*, 1999. 6

[6] J. Boreczky and L. Rowe. Comparison of video shot boundary detection techniques. *JEI*, 1996. 6

[7] M. Chen, S. Pang, T. Cham, and A. Goh. Visual tracking with generative template model based on riemannian manifold of covariances. In *ICIF*, 2011. 7

[8] Y. Feng, Y. Wu, and L. Fan. Online object reconstruction and tracking for 3D interaction. In *ICME*, 2012. 2

[9] K. Fragkiadaki, M. Salas, P. Arbelaez, and J. Malik. Grouping-based low-rank trajectory completion and 3D reconstruction. In *NIPS*, 2014. 3

[10] Y. Furukawa and J. Ponce. Accurate, dense, and robust multiview stereopsis. *PAMI*, 2010. 3

[11] R. Garg, A. Roussos, and L. Agapito. Dense variational reconstruction of non-rigid surfaces from monocular video. In *CVPR*, 2013. 3

[12] J. Heller, M. Havlena, M. Jancosek, A. Torii, and T. Pajdla. 3D reconstruction from photographs by CMP SfM web service. *MVA*, 2015. http://ptak.felk.cvut.cz/sfmservice/. 7, 8

[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 *BMVC*, 2012. 2, 6

[14] B. Horn and B. Schunck. Determining optical flow. *AI*, 1981. 7

[15] R. Jones, P. Haufe, E. Sells, P. Iravani, V. Olliver, C. Palmer, and A. Bowyer. RepRap – the replicating rapid prototyper. *Robotica*, 2011. 1

[16] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *SGP*, 2006. 6

[17] M. Kazhdan and H. Hoppe. Screened poisson surface reconstruction. *TOG*, 2013. 6

[18] M. Keller, D. Lefloch, M. Lambers, S. Izadi, T. Weyrich, and A. Kolb. Real-time 3D reconstruction in dynamic scenes using point-based fusion. In *3DV*, 2013. 2

[19] C. Kerl, J. Sturm, and D. Cremers. Robust odometry estimation for RGB-D cameras. In *ICRA*, 2013. 2

[20] A. Kundu, K. Krishna, and C. Jawahar. Realtime multibody visual SLAM with a smoothly moving monocular camera. In *ICCV*, 2011. 2

[21] K. Lebeda, S. Hadfield, and R. Bowden. 2D or not 2D: Bridging the gap between tracking and structure from motion. In *ACCV*, 2014. 2, 3, 7

[22] W. Lorensen and H. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *ACM CG*, 1987. 6

[23] R. Mur-Artal and J. Tardós. Fast relocalisation and loop closing in keyframe-based SLAM. In *ICRA*, 2014. 2

[24] R. Newcombe, S. Lovegrove, and A. Davison. DTAM: Dense tracking and mapping in real-time. In *ICCV*, 2011. 2

[25] M. Niessner, M. Zollhfer, S. Izadi, and M. Stamminger. Real-time 3D reconstruction at scale using voxel hashing. *TOG*, 2013. 2

[26] Q. Pan, G. Reitmayr, and T. Drummond. ProFORMA: Probabilistic feature-based on-line rapid model acquisition. In *BMVC*, 2009. 2, 6

[27] D. Ross, J. Lim, R.-S. Lin, and M.-H. Yang. Incremental learning for robust visual tracking. *IJCV*, 2008. 7

[28] C. Scheidegger, S. Fleishman, and C. Silva. Triangulating point set surfaces with bounded error. In *SGP*, 2005. 6

[29] S. Song and M. Chandraker. Joint SFM and detection cues for monocular 3D localization in road scenes. In *CVPR*, 2015. 3

[30] W. Tan, H. Liu, Z. Dong, G. Zhang, and H. Bao. Robust monocular SLAM in dynamic environments. In *ISMAR*, 2013. 2

[31] N. Tisserand and N. Burrus. Skanect, 3D scanning software. http://skanect.occipital.com/. 2

[32] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. DeepFlow: Large displacement optical flow with deep matching. In *ICCV*, 2013. 4

[33] T. Wiemann, H. Annuth, K. Lingemann, and J. Hertzberg. An extended evaluation of open source surface reconstruction software for robotic applications. *JIRS*, 2015. 6