SurfaceView: Seamless and tile-based orthomosaics using millions of street-level images from vehicle-mounted cameras

Supannee Tanathong, William A. P. Smith, and Stephen Remde

***Abstract*—We tackle the problem of building city- or country-scale seamless mosaics of the road network from millions of street-level images. These "orthomosaics" provide a virtual top-down, orthographic view, as might be captured by a satellite though at vastly reduced cost and avoiding limitations caused by atmospheric interference or occlusion by tree cover. We propose a novel, highly efficient planar visual odometry method that scales to millions of images. This includes a fast search for potentially overlapping images, relative pose estimation from approximate ground plane projected images and a largescale optimisation, which we call motion-from-homographies, that exploits multiple motion, GPS and control point priors. Since even city-scale orthomosaics have petapixel resolution, we work with a tile-based mosaic representation which is more efficient to compute and makes web-based, real-time interaction with the images feasible. Our orthomosaics are seamless both within tiles and across tile boundaries due to our proposed novel variant of gradient-domain stitching. We show that our orthomosaics are qualitatively superior to those produced using state-of-the-art structure-from-motion output yet our pose optimisation is several orders of magnitude faster. We evaluate our methods on a dataset of 1.4M images that we collected.**

***Index Terms*—Structure-from-motion, image stitching, gradient-domain blending, image mosaicing, tile-based mapping**

I. Introduction

BUILDING city- or country-scale maps of the highways has traditionally been the job of cartographers. Over the past two decades, the availability of planet-wide, high resolution satellite imagery has allowed these hand-built maps to be augmented by digital imagery and elevation data enabling a whole host of new applications. However, acquiring satellite imagery is hugely expensive, dependent on clarity of the atmosphere and resolution is limited by the large camera-surface distance. For applications such as road condition surveying, asset management and autonomous driving there is a need to obtain much higher resolution imagery of the highways with high temporal frequency so that deterioration and change can be detected and monitored. This motivates the use of street level imagery.

In this paper we describe a system for building virtual top-down view maps of very large areas of the highways simply by driving around a vehicle and collecting images from a camera with an oblique view of the road surface (e.g. see Figure 1, middle). The resulting images are referred to as "orthomosaics", due to the fact that they are a mosaic of many images (perhaps millions) and that they approximate an orthographic view of the world from above, as might be captured by a satellite. However, unlike satellite imagery, our system is vastly less costly, is not dependent on having a sufficiently clear atmosphere and the resolution of the images is limited only by the resolution of the cameras mounted on the vehicle. Hence, our orthomosaics allow visualisation of the road network from a scale where the entire network is visible, down to a scale where features such as surface cracks are resolved in very high detail. Fig. 1 provides an overview of our system and a video visualisation is available[1].

Our system provides an attainable approach to build street-level orthomosaics. Using only a GPS receiver and a camera mounted on top of the vehicle with an oblique view to the ground, when driving along the road, it captures images of the road surface at the highest resolution and steers clear of occlusions occupying the road surface unlike that collected by airborne devices. This results in artefact-free, high-resolution mosaics. Due to its practical setup, this can be used to encourage voluntary crowd-sourcing street-level images for community benefits. In addition, our survey driving-based system is practical while, in some countries, flying drones/UAVs in cities or residential areas are prohibited by law.

While methods for 3D modelling, i.e. structure-from-motion (SfM), have been developed for very large-scale, street level image datasets [1], the creation of seamless orthomosaics has not previously been considered at this scale. While SfM reconstructs sparse 3D models (each image contributes on the order of a thousand feature points to the 3D scene), our goal is to produce orthomosaics at the maximum resolution possible from the input images (hence, each image contributes on the order of a million pixels to the output mosaic). Hence, our problem is roughly three orders of magnitude larger in scale than a sparse 3D reconstruction.

Contribution In this paper, we propose specialised approaches towards creating large-scale orthomosaics at the scale and quality needed for country-wide road surface inspection. Our system comprises the pipeline shown in Fig. 2 with two key novelties. First, we propose an alternative method to SfM to solve for camera pose. With the assumption of a locally planar surface, we reduce the complexity of the problem to solve for only 6 unknowns per image and eliminate

S. Tanathong and S. Remde are with Gaist Solutions Ltd. Email: supannee.tanathong@gmail.com, stephen.remde@gaist.co.uk

W. Smith is with the Department of Computer Science, University of York, York, UK. Email: william.smith@york.ac.uk

[1]https://youtu.be/tFftnwe7ncE

Fig. 1. Overview of our system. We build city scale, virtual top down view mosaics from street-level images at a scale where the entire network is visible down to a scale where fine details of road surface can be inspected. See supporting video[1].

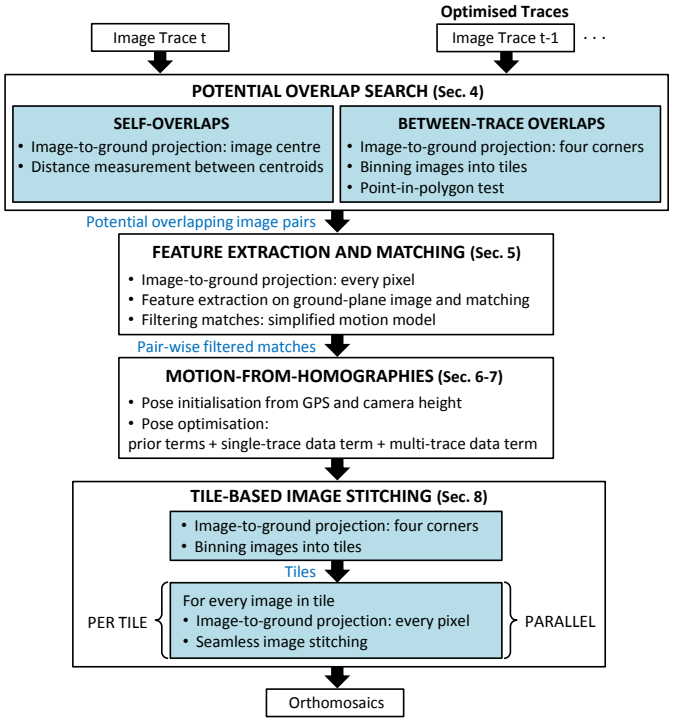

Fig. 2. Our proposed pipeline for creating large-scale orthomosaics.

the need for computing 3D scene points while achieving the same accuracy as that obtained by conventional SfM (or better where the planar degeneracies of a road scene cause SfM to fail). Second, we propose a tile-based image stitching method that can create seamless orthomosaics from millions of images while preserving finescale details. Every step of our pipeline scales to extremely large datasets, including a fast method for finding potentially overlapping images and simplified transformation model for filtering feature matches.

The rest of the paper is organised as follows: Sec. II reviews related work. Sec. III defines notations and basic models used in the later sections. Sec. IV-VIII describe the complete pipeline as presented in Fig. 2. We demonstrate the results of our method on our own dataset of 1.4M images in Sec. IX including comparison with existing methods. Finally, conclusions and future work are discussed in Sec. X.

II. Related Work

This paper presents a system for building orthomosaics from a collection of street-level images captured by a vehicle-mounted camera with an oblique view of the road surface. We see three categories of related work.

Camera pose estimation Prior to generating image mosaics of a scene, the poses (positions and orientations) of the cameras must be known. Main techniques used to obtain the poses include simultaneous localization and mapping (SLAM) and structure-from-motion (SfM).

SLAM These methods concurrently observe the surroundings to localise the observer's position while incrementally building an accurate map of the environment. Approaches in SLAM can generally be classified based on the type of sensors used to sense the environment. LiDAR SLAM uses LiDAR sensors at its core while visual SLAM uses cameras as a main sensor such as monocular-camera [2]–[6], stereo [7]–[9], RGB-D [8], [10], [11], etc. Monocular-based approaches, although widely adopted due to their simple and economical setup, suffer from recovering the metric scale, and that integrating with an IMU (either loosely-coupled or tightly-coupled), referred to with a prefix *visual-inertial*, offers a main advantage of solving scale ambiguity and providing a more robust navigation. Examples of work include [12]–[15]. Visual-based approaches may be classified, based on how image information is utilised, as direct-based [16], for dense reconstruction, or feature-based approaches [5], for sparse reconstruction. The majority of SLAM methods tackle the navigation problem in real-time, thus imposing a constraint on computational resources. The approaches often require high frame-rate image-sequences, although some report to be able to work on a low frame-rate [4]. Many SLAM methods are key-frame based approaches [7], [12], [17], in which key-frames (subset of the entire images) are used to build the maps hence reduce the scale of the optimisation tasks. Many works have claimed to solve large-scale problems [4] (reported on several kilometre), [7] (thousands of keyframes), [17] (several hundred metres), also [5], [7], [9], [18], still not at the level large-scale SfM methods have achieved. For a comprehensive review of SLAM, see [19]–[21].

SfM Given a set of images, SfM estimates the poses of the camera and simultaneously reconstructs a sparse 3D model of the scene. Many existing SfM approaches have been proposed

and can be broadly classified as incremental or global based on how images are registered into the reconstructed model. Incremental SfM [22]–[25] starts by constructing an initial coarse model from a small number of images then iteratively adds more images to grow the model. The incremental process repeatedly uses bundle adjustment to re-estimate camera poses and refine the model. Although the technique has been shown to be successful, especially in solving very large scale problems such as Internet photo datasets [26]–[28], it tends to be computationally expensive and may suffer from accumulated errors [29].

Global SfM [30]–[32] on the other hand, estimates camera parameters of all images simultaneously which leads to efficient computation and encourages scalability. A key success factor of the global approach lies on a good initialisation of camera orientations and positions for bundle adjustment. While approaches to estimating global rotation of each camera are well established [33]–[35], computing global translation priors is challenging, but recent studies have shown promising results [36], [37]. Moreover, incorporating extra information, such as sparse depth [30], vanishing points [38], GPS [39]–[42] and inertial data, has proven useful with the latter two common for photogrammetry and SLAM.

SfM approaches may also be divided into structure-based or structure-less. The former refers to the conventional techniques in which 3D structures are estimated simultaneously with camera poses. While the latter eliminates 3D structures from the bundle adjustment and is not so widely used. Instead of minimising the cost of reprojection errors, Rodrigue et al. [43] formulate their cost function based on the pairwise epipolar constraints in which unknowns include only camera poses of the two views. Similarly, Indelman et al. [44] exclude structures and form the cost function based on multi-view constraints. Zhou et al. [45] estimate camera parameters for planar scene by minimising reprojection error based on homography. Lovegrove et al. [46] perform visual odometry also using a homography model with a global image alignment cost. Sawhney et al. [47] optimise a similar cost function to ours for small scale mosaicing.

Image Stitching After images are globally aligned through SfM, in order to create pleasant looking mosaics the image contents must be blended seamlessly through a technique called image stitching. Image stitching can be classified as transition smoothing or optimal seam approaches. Most recent transition smoothing algorithms combine images in the gradient domain. Instead of copying image intensities, Perez et al. [48] copy the gradient field of the source region onto the target region then the gradients in the target image are obtained by solving a linear equation system constructed as Poisson equation with Dirichet boundary served as guidance field. Levin et al. [49] create the composite image by optimising the dissimilarity between the composite image and the source images over the gradient domain. The optimal seam finding approaches place the seams to where disagreements in the composite images are hardly noticeable. The most commonly used technique for seam placement is through graph-cuts [50], [51], other techniques include watershed segmentation [52] and dynamic programming [53], [54]. The combination of transition smoothing and optimal seam approaches is presented in [51], where the source images are composed along the seams then gradient-domain fusion is applied to reduce photometric inconsistencies. Recent studies have shown that deep learning can also be used for image compositing [55], [56].

Solving a system of linear equations becomes computationally expensive and requires huge memory especially when the final composite image is large. Agarwala [57] reduce the size of the problem by blending only where seams are present by adaptively subdividing the image domain using a quadtree hierarchy; reducing the problem by solving for the size of the seam pixels instead of the size of the composite image which substantially speeds up the computation. However, when the number of source images increases or there are too many optimal seams (e.g. small overlapping regions) the performance can become poorer. An efficient computation can also be achieved by implementing the gradient-domain solver in a distributed manner as presented in [58].

Large-scale Image Mosaics Large-scale, high resolution image mosaics are a useful representation for many analysis and mapping tasks. Koef et al. [59] present a system to create seamless "gigapixel" images and store the resulting panoramic images as a pyramid of fixed-sized tiles which are fetched to show only when requested. Prados et al. [60] create "giga-mosaics" of large areas of seabed from sequences of images with varying ligthing and altitudes. Ferrer et al. [61] used similar priors to ours in the context of creating large-scale underwater mosaics. We find the method to create photo-mosaics in [60] to be similar to ours, although much different in implementation details, in which (a) image stitching is independently operated on tiles but still ensures smooth colour transition between neighbour tiles (b) stitched images preserve richness detail of the scene. However, ours is a single-step process, while a post processing is required by [60] for smooth transitions between adjacent tiles.

Many recent studies have shown impressive results in generating large-scale photo-mosaics from UAV images. Map2dfusion [62] builds large-scale orthomosaics in real-time by incrementally blending warped image patches using an adaptive weighted multi-band algorithm. Because the mosaic plane is estimated by fitting sparse point clouds, the orthomosaics are not guaranteed to be completely planar. By assuming a UAV flying orthogonal to the ground plane, MGRAPH [63] discounts the roll and pitch motion such that the captured images are restricted to a similarity transformation. To achieve real-time performance, overlaying images are not blended together, leading to visual artifacts under different lighting conditions. OpenREALM [64] presents a system for real-time aerial mapping system which can generate orthomosaics from UAVs with a downward facing camera. Due to no exposure correction or colour blending, the generated mosaics contain artefacts around boundary of individual images. There are additional work on large-scale mosaics in different settings such as MAV [65], ROV and AUV [66] and commercial

products such as Agitosft Photoscan[2] and Context Capture[3].

III. Preliminaries

We assume that a sequence of images (which we refer to as a "trace") is captured by a moving vehicle with a fixed camera oriented towards the road surface. A complete dataset comprises N_t traces, where N_t could be $> 1,000$. We denote by $I_c^{i,t}(x,y)$ the intensity in colour channel c in the ith image in trace t at pixel position (x,y), with $i \in \{1,\ldots,N_t\}$, $x \in \{1,\ldots,W^{\text{captured}}\}$ and $y \in \{1,\ldots,H^{\text{captured}}\}$. We assume that the camera in trace t is calibrated, i.e. that the intrinsic matrix $\mathbf{K}_t$ is known, where $f_{x,t}$ and $f_{y,t}$ are the focal lengths in the x and y direction and $c_{x,t}$ and $c_{y,t}$ define the centre of projection, and that the nonlinear distortion parameters, $k_{1,t}$, $k_{2,t}$, $p_{1,t}$ and $p_{2,t}$, are also known. We assume that the height of the camera above the road, w_t^{measured}, and the angle that the camera view vector makes with the road surface, $\beta_t^{\text{measured}}$, are approximately known, though they are later optimised per-image so these measured values are used only for initialisation.

A. Motion model

We represent the pose of the camera captured the ith image in trace t by the rotation matrix $\mathbf{R}_{i,t} \in SO(3)$ and the translation vector $\mathbf{t}_{i,t} \in \mathbb{R}^3$. Where we refer to only a single trace, we drop the dependency on t.

A world point is represented by coordinates $\mathbf{w} = [u,v,w]^T$, where (u,v) is a 2D Universal Transverse Mercator (UTM) coordinate representing position on the $w = 0$ ground plane and w is altitude above sea level. Each camera has a standard right handed coordinate system with the optical axis aligned with the w axis. A world point in the coordinate system of the ith camera is given by

$$\mathbf{w}_i = \mathbf{R}_i\mathbf{w} + \mathbf{t}_i. \tag{1}$$

It is convenient to represent the rotation as a composition of four rotation matrices, one of which is fixed. The fixed one aligns the world w axis with the optical axis of the camera in canonical pose: $\mathbf{R}_{\text{w2c}} = \mathbf{R}_x(90°)$. We model vehicle orientation by three angles: yaw, pitch and roll. We choose this representation because the vehicle motion model leads to constraints that can be expressed naturally in terms of these angles. Hence, we define three rotation matrices $\mathbf{R}_{\text{yaw}}(\alpha)$, $\mathbf{R}_{\text{pitch}}(\beta)$ and $\mathbf{R}_{\text{roll}}(\gamma)$ for which the yaw angle rotating around the camera's y axis, the pitch angle around the x axis and the roll angle around the z axis.

The overall rotation as a function of these three angles is given by:

$$\mathbf{R}(\alpha,\beta,\gamma) = \mathbf{R}_{\text{roll}}(\gamma)\mathbf{R}_{\text{pitch}}(\beta)\mathbf{R}_{\text{yaw}}(\alpha)\mathbf{R}_{\text{w2c}}. \tag{2}$$

Hence, the rotation of the ith camera depends upon the estimate of the three angles for that camera:

$$\mathbf{R}_i = \mathbf{R}(\alpha_i,\beta_i,\gamma_i). \tag{3}$$

The coordinate systems and rotation angles are visualised in Fig. 3. We assume that each image is labelled with a geotag, $(u_i^{\text{GPS}}, v_i^{\text{GPS}})$, providing an approximate position for the camera in the ith image in world coordinates. In our system, this geotag is provided by GPS augmented by wheel tick odometry.

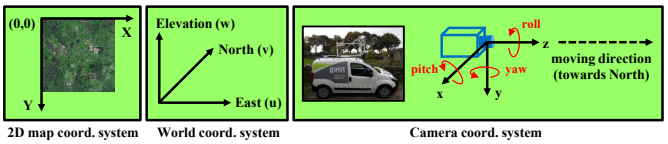

Fig. 3. The three coordinate systems as referred to in this paper.

B. Perspective projection and distortion

We assume that our cameras can be modelled by a pin-hole perspective projection followed by radial and tangential distortion. Accordingly, we define a projection matrix as

$$\mathbf{P} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\\ \mathbf{0}_{1\times 3} & 1\end{bmatrix}, \tag{4}$$

and write the perspective projection of a world point $\mathbf{w}$ as

$$\mathbf{x}' = h\left(\mathbf{P}\begin{bmatrix}\mathbf{w}\\ 1\end{bmatrix}\right). \tag{5}$$

where the function $h : \mathbb{R}^3 \to \mathbb{R}^2$ homogenises a 3D point:

$$h([x,y,z]^T) = \left[x/z, y/z\right]^T. \tag{6}$$

For world points lying on the $w = 0$ ground plane, i.e. $[u,v,0]^T$, we can express the transformation from ground plane to image plane via the following homography:

$$\mathbf{H} = \mathbf{K}\begin{bmatrix}\mathbf{R}\mathbf{S}^T & \mathbf{t}\end{bmatrix}, \quad \mathbf{S} = \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\end{bmatrix} \tag{7}$$

such that

$$\mathbf{x}' = h\left(\mathbf{H}\begin{bmatrix}u & v & 1\end{bmatrix}^T\right). \tag{8}$$

The inverse projection from undistorted image coordinate (x,y) to the ground plane is simply:

$$\begin{bmatrix}u\\ v\end{bmatrix} = h\left(\mathbf{H}^{-1}\begin{bmatrix}x & y & 1\end{bmatrix}^T\right). \tag{9}$$

To obtain pixel coordinates, we apply the distortion model of Heikkilä and Silven [67] to the projected point:

$$\mathbf{x} = \text{distort}(\mathbf{x}', f_x, f_y, c_x, c_y, k_1, k_2, p_1, p_2) \tag{10}$$

Note that inverting the distortion function involves solution of a nonlinear optimisation problem so, for efficiency, we avoid undistorting image coordinates wherever possible.

C. Image to ground plane mapping

As illustrated in Fig. 1, our images provide an oblique perspective view of the road surface (Fig. 1 middle). We ultimately wish to produce a top-down, orthographic mosaic of all images in a dataset. Therefore, during the final image stitching, and also during feature extraction and matching, we wish to project and resample captured images to the ground plane. We do this using backwards mapping. This is more efficient since it amounts to bilinear interpolation of a regularly

[2]https://www.agisoft.com
[3]https://www.bentley.com/en/products/brands/contextcapture

sampled function (the input image) which is more efficient than barycentric interpolation of an irregularly sampled function (which would be required if forward mapping was used). Moreover, we avoid having to undistort the pixels in the original image (using a backwards mapping we only need compute a forward distortion).

Concretely, we define a grid on the ground plane in terms of UTM coordinates $u(X,Y)$ and $v(X,Y)$ where (X,Y) is a pixel coordinate in the ground plane image with $X \in \{1,\dots,W^{\text{grid}}\}$, $Y \in \{1,\dots,H^{\text{grid}}\}$. Given an estimated camera pose, the ground plane grid coordinates, $[u(X,Y), v(X,Y)]$, are projected to the image plane via (8), then distorted via (10) giving an image coordinate (x,y) that can be used for bilinear interpolation into $I_c^{i,t}(x,y)$ and the resulting value assigned to $I_c^{\text{grid}}(X,Y)$. This process is repeated for all ground plane image pixels (X,Y).

D. Tile-based mapping system

We use a multi-resolution, tile-based mapping system to store and process our orthomosaics. Our implementation follows Google Maps conventions in which the spherical Mercator projection is used and the size of a tile is $H^{\text{tile}} = W^{\text{tile}} = 256$ pixels [68]. Accordingly, the width and height of the map at a zoom level Z is 256×2^Z pixels, so at zoom level $Z = 0$ the entire Earth is covered by a single tile. At higher zoom levels, tiles within the map are indexed via their row, R, and column, C, with $R, C \in \{1,\dots,2^Z\}$. Hence, any tile can be uniquely indexed by $Z/C/R$. We denote by $I_c^{Z/C/R}(X,Y)$ colour channel c of pixel (X,Y) in the tile $Z/C/R$, with $X, Y \in \{1,\dots,256\}$.

E. Binning images into tiles

Both for finding potential overlapping images between traces (Sec. IV-B) and for stitching orthomosaic tiles (Sec. VIII), it is helpful to associate with each tile the set of images that contain parts of the ground plane that overlap the tile. Depending on the camera parameters, multiple images may overlap a single tile. We denote the set of images whose ground plane projection overlaps tile $Z/C/R$ by $\mathcal{P}(Z,C,R)$, where $(i,t) \in \mathcal{P}(Z,C,R)$ if image i in trace t overlaps tile $Z/C/R$. Computing these sets is a spatial binning problem and thus runs in linear time [69].

We project image corners onto the ground plane using (9) to form a 4-sided polygon, where the homography depends on the estimated camera pose $\mathbf{R}_{i,t}$ and $\mathbf{t}_{i,t}$ and results in images projecting to different locations on the ground plane. An image is binned into a tile if the ground plane projection of any of the four image corners lies within the tile boundaries. As camera lenses are subject to distortion, using distorted image coordinates in (9) may omit some images near the border of tiles. On the other hand, deriving undistorted coordinates from distorted coordinates is not computationally efficient. A simple solution is to extending the tile boundary by a few pixels to compensate for image distortion.

IV. Fast search for potential overlaps

The first step in our pipeline uses fast geometric heuristics to identify pairs of images that may contain overlapping regions of the ground plane. The goal of this part of the pipeline is to reduce the number of image pairs for which we must conduct expensive image feature matching and match filtering. We use two different procedures to identify potential overlaps between: 1. images from the same trace (which we refer to as self-overlaps) and 2. images from different traces (which we refer to as between-trace overlaps).

A. Self-overlaps

Self-overlaps are detected for each trace independently according to two heuristics. First, our images come from a sequence. Moreover, in our setup we trigger image capture using GPS and wheel tick odometry information such that there is an approximately fixed distance between consecutive images. This means that it is possible to choose a constant offset $O \in \mathbb{N}_{>0}$, within which we expect images to overlap, i.e. we expect image i to contain overlaps with images $i-O$, $\dots$, $i+O$. We refer to this as a sequence heuristic and define the set of such potentially overlapping pairs for trace t as:

$$\mathcal{O}^t_{\text{seq}} = \{(i,j) | 1 \leq i,j \leq N_t \wedge 0 < j - i \leq O\}, \tag{11}$$

and hence $|\mathcal{O}^t_{\text{seq}}| = N_t O - O(O+1)/2$.

Second, since the trajectory of the capture vehicle may intersect with itself or even retrace previously covered ground, we also seek to detect potential overlaps between images that are not close in the image sequence. We do this by comparing the ground plane projection of image centres. The distance between projected image centres is given by:

$$d_{\text{centroid}}(i,j) = \left\| \mathbf{H}_i^{-1} \begin{bmatrix} c_x \\ c_y \\ 1 \end{bmatrix} - \mathbf{H}_j^{-1} \begin{bmatrix} c_x \\ c_y \\ 1 \end{bmatrix} \right\| \tag{12}$$

where $\mathbf{H}_i$ is the homography that transforms from the ground plane to the ith image computed according to (7). Of course, the homography depends upon camera pose estimates which we have not yet optimised. Hence, for this test we construct the homography using the initialisation from GPS location and bearing as described in Sec. VI-A.

We search for all pairs where the Euclidean distance between centroids is less than a threshold:

$$\mathcal{O}^t_{\text{centroid}} = \{(i,j) | d_{\text{centroid}}(i,j) < t\} \tag{13}$$

This is a 2D fixed-radius all nearest neighbours search, which can be computed in $O(N_t + |\mathcal{O}^t_{\text{centroid}}|)$ time [70]. The set of potentially self-overlapping image pairs is then given by:

$$\mathcal{O}^t_{\text{self}} = \mathcal{O}^t_{\text{seq}} \cup \mathcal{O}^t_{\text{centroid}}. \tag{14}$$

B. Between-trace overlaps

During pose estimation, we begin by optimising over each trace independently (see Sec. VI) prior to incrementally combining traces (see Sec. VII). Hence, when searching for potential overlaps between images from different traces, we have already performed pose optimisation on individual traces

and so pose estimates $(\mathbf{R}_{i,t}, \mathbf{t}_{i,t})$ are available for every image in every trace. If trace s and t contain potentially overlapping images, then we store a set, $\mathcal{O}^{s,t}_{\text{between}}, s < t$, such that $(i,j) \in \mathcal{O}^{s,t}_{\text{between}}$ if image i in trace s is a potential overlap for image j in trace t. To compute this set, we proceed through a sequence of tests of increasing complexity so that the number of images considered by the more expensive and accurate tests is kept to a minimum.

To find candidate pairs from the entire image set, we use the spatial binning process described in Sec. III-E. Any tile that contains images from more than one trace could contain images with between-trace overlaps. For all pairs of images from different traces that are binned to the same tile, we now test for intersections between the bounding boxes of the ground plane projections of the images. If the bounding boxes overlap, we perform a point in polygon test [71]. Two images are considered potentially overlapping if (1) the centroid of one polygon is inside the other, or (2) at least 3 corners of one polygon are inside the other, or (3) two corners are inside the other and those are not adjacent, or (4) at least 1 corner is inside the other and the area of overlap is greater than a threshold. For the latter case, we need to compute the area of overlap. First we form a polygon from the corners inside the other polygon and the edge intersections. Then the area is efficiently computed using Meister's shoelace formula [72].

V. Feature Extraction and Matching

Having found pairs of images that potentially contain overlapping regions of the ground plane, we then proceed to compute feature matches between images. We make this process robust by filtering the matches according to a simplified motion model. Any pair of images with an insufficient number of inlying matches is discarded. The feature matches between all other pairs are retained and used later in the pose optimisation process. Unlike in a conventional 3D SfM pipeline, we compute only pairwise matches. This is because we require only pose estimates and do not reconstruct the 3D position of matched features. This vastly reduces the complexity of the pose optimisation process while being sufficient for our final goal of orthomosaic stitching.

The part of the scene in which we are interested (the road surface) can be assumed to be locally planar. We exploit this during feature matching in two novel ways. First, we perform feature extraction on images that have been reprojected to an assumed ground plane. This has the effect of normalising the appearance of features that lie on the ground plane (by approximately undoing the effect of perspective projection) and improving the likelihood of matching the same point viewed from different locations. It also has the added benefit of distorting the appearance of features that do not lie on the road plane (such as buildings, signage or vehicles) so that they are less likely to be matched. Second, we assume a simplified motion model with only three degrees of freedom which means that feature matches can filtered efficiently and robustly.

A. Image to ground plane mapping

We map each image to a canonical ground plane projection using the method described in Sec. III-C under an assumed pose $\mathbf{c} = [0, 0, w^{\text{measured}}]^T$, $\alpha = \gamma = 0$ and $\beta = \beta^{\text{measured}}$. Any deviations in $\mathbf{c}_w$ or β from their measured values or γ from zero will cause this projection to be distorted but we expect these deviations to be small and the distortions to be insignificant for this part of the process.

Specifically, to define the ground plane grid, we project the image corners assuming canonical pose using (9) and compute a bounding box. We then define a regular grid of specified resolution over the extent of this bounding box. As in Sec. III-C, we project each grid point back into the image using (8) and perform forward distortion with (10). Note that the grid and its projection depend only upon w^{measured}, β^{measured}, $\mathbf{K}$ and the nonlinear distortion parameters. Therefore the grid and its projection only need be computed once per trace. For each image, we now perform bilinear interpolation at the grid sample points to compute a ground plane projected image. An example of two input images and their ground plane projections can be seen in rows 1 and 2 of Fig. 4.

B. Feature matching

We compute local features in the ground plane images. Let $\mathbf{u}_f$ denote the location in the ground plane image of feature f. We also compute and store $\mathbf{x}'_f$, by projecting $\mathbf{u}_f$ back into the image with (8) (note that we do not distort this location since we will later project them back to the ground plane during pose optimisation). The $\mathbf{x}'_f$ are the undistorted locations of features in the image plane.

We expect features in the projected ground plane images to approximately retain their appearance and scale but that the orientations of the same features in two different images will differ by a global 2D rotation. Hence, we do not require invariance to affine distortion but do require rotational invariance at the initial matching stage. While there are a number of feature descriptors that satisfy these criteria, we use SIFT [73] though any other suitable descriptor could be substituted. We compute greedy matches between features in pairs of images identified as potentially overlapping using the methods in Sec. IV. We filter these matches for distinctiveness using Lowe's ratio test [73].

Even with this filter applied, the matches will still contain noise that will disrupt the alignment process. Specifically, they may include matches between features that do not lie on the road plane (such as buildings or signage) or between dynamically moving objects (such as other vehicles). If such matches were retained, they would introduce significant noise into the pose refinement process.

C. Filtering matches with a simplified motion model

We now filter feature matches such that they are geometrically consistent with a simplified motion model. Specifically, we assume that between a pair of images, the vehicle may translate in the u, v plane and that its bearing may change. In other words, we assume that pitch, roll and the height of the camera above the ground plane remain fixed. This restricted transformation therefore only has three degrees of freedom. In the projected ground plane images, this means that we seek a 2D proper rigid transformation (i.e. a translation and

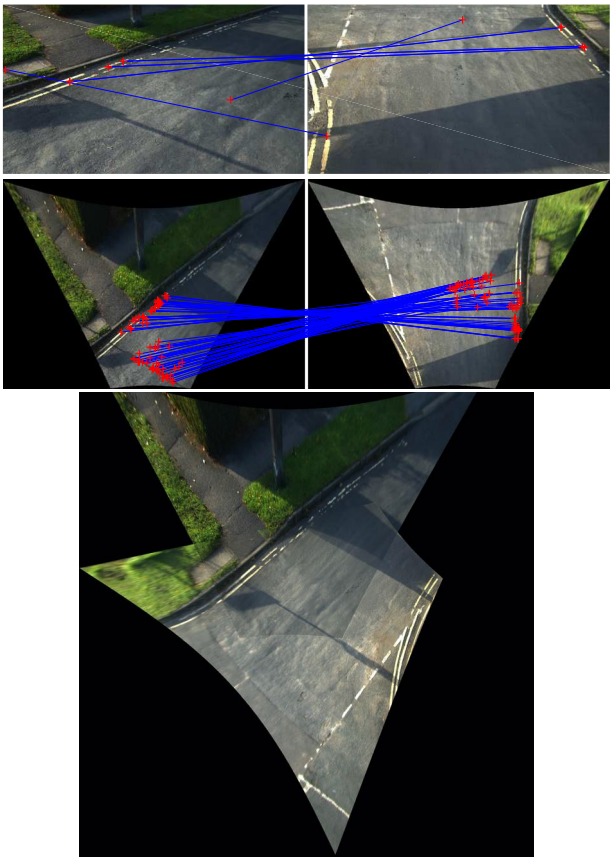

Fig. 4. Feature matching in the ground plane. First row (naive method): feature matches on undistorted images that are consistent with a homography. Second row (proposed method): feature matches in images projected to ground plane that are consistent with 2D proper rigid transformation. Bottom: alignment using matches in second row.

rotation). The least squares best fitting such transformation can be found in closed form using a restricted version of Procrustes alignment [74] (allowing only translation and rotation). To measure the error between a pair of matched feature locations after computing a best fitting transformation, we use the Procrustes distance and symmetrise by taking an average of the forward/backward transformation.

We use this simplified motion model in conjunction with RANSAC to compute the set of inlying feature matches. From the greedy feature matches, we randomly select two matches (the minimum required to fit a 2D proper rigid transformation). We fit the transformation and define inliers as those with a Procrustes distance of < 10 pixels. An example of the final result is shown in Fig. 4. In the top row we show a pair of input images. The feature matches after RANSAC filtering between features in the projected ground plane images are shown in the second row. For comparison, the top row also shows the result of matching directly in the original images (we first undistort the images, compute greedy feature matches, then fit a general homography to filter the matches). It is clear that the ground plane images provide more matches than the naive method and that they correspond to the correct transformation. Finally, at the bottom of the figure we show the alignment using the fitted 2D proper rigid transformation. Note that this is not the alignment used in the final orthomosaic stitching. It is just the coarse transformation used to filter feature matches.

The final set of inlying matches are stored as $\mathcal{M}_{i,j}^{s,t}$, where $(f, g) \in \mathcal{M}_{i,j}^{s,t}$ if feature f in the ith image in trace s is matched to feature g in the jth image of trace t. For the set of matches between images in the same trace, we write simply $\mathcal{M}_{i,j}^{t}$. The locations of the features in the ground plane projected images are not used in the following pose optimisation and can be discarded at this point.

VI. Motion-from-homographies

We now address the problem of estimating accurate camera poses for each image in a single trace. The pose needs to be sufficiently accurate that, when images are later projected to the ground plane, overlapping images have at least pixel-accurate alignment. Otherwise, there will be misalignment artefacts in the stitched orthomosaic images. This can be viewed as an SfM problem. However, previous approaches for SfM are not applicable in this setting for two reasons: **1.** Images in which the scene is primarily the road surface are largely planar. This is a degenerate case for estimating a fundamental matrix and subsequently reconstructing 3D scene points. **2.** Typically, the number of 3D scene points reconstructed by the SfM process is much larger than the number of pose parameters to be estimated. This means that SfM does not scale well to very large problems, e.g. those containing millions of images.

Similarly, methods based on visual odometry or Simultaneous Localisation and Mapping (SLAM) are not applicable. They typically require high frame rates in order to robustly track feature points over time. Sampling images at this frequency is simply not feasible when we wish to build orthomosaics of thousands of kilometres of road.

The pipeline described in Sec. IV and V provide sets of matched features between pairs of images within the same trace and between different traces. These matches have been found and filtered by assuming that the road surface is locally planar so that images can be projected to the ground plane via a homography. For this reason, we refer to our method as motion-from-homographies. We now solve an optimisation problem that uses the estimated pairwise matches directly without reconstructing the 3D world position of the features. This vastly reduces the complexity of the optimisation problem that we need to solve compared to SfM. The number of unknowns is simply $6N$ for an N image sequence.

A. Initialisation

We rely on GPS and an initial estimate of the camera height above the road surface to initialise the location of each camera: $\mathbf{c}_\mathbf{i}^{\text{init}} = [u_i^{\text{GPS}}, v_i^{\text{GPS}}, w^{\text{measured}}]^T$ where $(u_i^{\text{GPS}}, v_i^{\text{GPS}})$ is the GPS estimate of the ground plane position of the ith camera and w^{measured} is the measured height of the camera above the road surface in metres. This need only be a rough estimate as the value is subsequently refined.

To initialise rotation, we compute yaw from the GPS bearing. First, we compute a bearing vector using central

differences: $\mathbf{b}_i = 0.5[u^{\text{GPS}}_{i+1} - u^{\text{GPS}}_{i-1}, v^{\text{GPS}}_{i+1} - v^{\text{GPS}}_{i-1}]^T$. Second, we convert this into a yaw angle: $\alpha^{\text{init}}_i = \text{atan2}(-\mathbf{b}_{i,1}, \mathbf{b}_{i,2})$.

We initialise the pitch to a measured value for the angle between the camera optical axis and the road surface and the roll to zero: $\beta^{\text{init}}_i = \beta^{\text{measured}}$ and $\gamma^{\text{init}}_i = 0$. Again, β^{measured} only need be roughly estimated since it is later refined.

B. Single trace data objective

For a single trace, we compute an objective function, $\varepsilon_{\text{data}}$, which measures how well matched features align in the ground plane. We refer to this as our data term:

$$\varepsilon^t_{\text{data}}(\{\mathbf{R}_i, \mathbf{t}_i\}_{i=1}^{N_t}) = \sum_{(i,j)\in\mathcal{O}^t_{\text{self}}} \sum_{(f,g)\in\mathcal{M}^t_{i,j}} \left\| h\left(\mathbf{H}_i^{-1}\begin{bmatrix}\mathbf{x}'_{f,i}\\1\end{bmatrix}\right) - h\left(\mathbf{H}_j^{-1}\begin{bmatrix}\mathbf{x}'_{g,j}\\1\end{bmatrix}\right)\right\|^2 \tag{15}$$

where $\mathbf{x}'_{f,i}$ is the 2D position in the image plane of the fth feature in image i, obtained by projecting the ground plane feature location back to the image, as described in Sec. V-B. $\mathbf{H}_i$ is the homography from the ground plane to the ith image given by (7) using the estimated $\mathbf{R}_i$ and $\mathbf{t}_i$. (15) shares some similarities with the data term used in bundle adjustment in the general SfM problem. However, there are some important differences. First, rather than measuring "reprojection error" in the image plane, we measure error when image features are projected to the ground plane. Second, the objective depends only on the camera poses - we do not need to estimate any 3D world point positions. The first difference is important because it encourages exactly what we ultimately want: namely, that corresponding image positions should align in the final orthomosaic. The second difference is important because it vastly reduces the complexity of the problem and makes it viable to process very large sets of images.

C. Priors

We not only wish to minimise the ground projection error, Sec. VI-B, we also wish to constraint the camera poses based on the *priors* of the vehicle. Since we expect the vehicle's orientation with respect to the road surface to remain approximately constant, we can impose priors on two of the angles. First, we expect side-to-side "roll" to be small, in general only being non-zero when the vehicle is cornering. Hence, our first prior simply penalises the variance of the roll angle estimates from zero:

$$\varepsilon_{\text{roll}}(\{\mathbf{R}_i, \mathbf{t}_i\}_{i=1}^{N_t}) = \sum_{i=1}^{N_t} \gamma_i^2. \tag{16}$$

The second angular prior penalises variance in the angle between the camera optical axis and the road plane, i.e. the pitch angle, such that it remains close to the average tilt angle:

$$\varepsilon_{\text{pitch}}(\{\mathbf{R}_i, \mathbf{t}_i\}_{i=1}^{N_t}) = \sum_{i=1}^{N_t}\left(\beta_i - \frac{1}{N}\sum_{i=1}^{N}\beta_i\right)^2. \tag{17}$$

Third, we penalise variance in the estimated height of the camera above the road surface since we expect this to remain relatively constant, hence the average height:

$$\varepsilon_{\text{height}}(\{\mathbf{R}_i, \mathbf{t}_i\}_{i=1}^{N_t}) = \sum_{i=1}^{N_t}\left(\mathbf{c}_{i,3} - \frac{1}{N}\sum_{i=1}^{N}\mathbf{c}_{i,3}\right)^2. \tag{18}$$

Fourth, we may be provided with *ground control points* (GCPs). These are a set of N_g points for which the ground plane coordinates $(u^{\text{GCP}}_1, v^{\text{GCP}}_1), \ldots, (u^{\text{GCP}}_{N_g}, v^{\text{GCP}}_{N_g})$ are known and the correspondence of these points to pixels in some images is known. GCP correspondences are stored in the set $\mathcal{C}_{\text{GCP}}$ such that $(i, j, (x, y)) \in \mathcal{C}_{\text{GCP}}$ if pixel (x, y) in image i corresponds to the jth GCP. We penalise the distance in the ground plane between the actual GCP position and that predicted by the pose of a camera which has a correspondence to a GCP:

$$\varepsilon_{\text{GCP}}(\{\mathbf{R}_i, \mathbf{t}_i\}_{i=1}^{N_t}) = \sum_{(i,j,(x,y))\in\mathcal{C}_{\text{GCP}}}\left\|\begin{bmatrix}u^{\text{GCP}}_j\\v^{\text{GCP}}_j\end{bmatrix} - h\left(\mathbf{H}_i^{-1}\begin{bmatrix}x\\y\\1\end{bmatrix}\right)\right\|^2.$$

Fifth, we introduce *ground control lines* (GCLs). In some cases, it is difficult to define a point-to-point correspondence between image and map but road boundaries, kerbs or road lines may allow us to associate an image point to a line on the ground plane. We define a set of N_l GCLs, $\mathcal{C}_{\text{GCL}}$, such that $(i, j, (x, y)) \in \mathcal{C}_{\text{GCL}}$ if pixel (x, y) in image i corresponds to the jth GCL. We define a GCL by its 2D line equation in the ground plane, such that the jth GCL is defined by: $a_j u + b_j v + c_j = 0$. We penalise the point-line distance between the GCL and the ground plane projection of pixels in $\mathcal{C}_{\text{GCL}}$:

$$\varepsilon_{\text{GCL}}(\{\mathbf{R}_i, \mathbf{t}_i\}_{i=1}^{N_t}) = \sum_{(i,j,(x,y))\in\mathcal{C}_{\text{GCL}}}\frac{1}{\sqrt{a_j^2+b_j^2}}\left|\begin{bmatrix}a_j\\b_j\\c_j\end{bmatrix}^T h\left(\mathbf{H}_i^{-1}\begin{bmatrix}x\\y\\1\end{bmatrix}\right)\right|.$$

Finally, we can use the estimated camera position provided by GPS in two ways. First, we can penalise absolute deviation from the GPS position:

$$\varepsilon_{\text{AGPS}}(\{\mathbf{R}_i, \mathbf{t}_i\}_{i=1}^{N_t}) = \sum_{i=1}^{N_t}\left\|\mathbf{c}_i - \mathbf{c}^{\text{init}}_i\right\|^2. \tag{19}$$

However, GPS can provide more accurate *relative* rather than absolute position information [75]. Hence, a second prior can penalise deviation from the relative position change provided by GPS:

$$\varepsilon_{\text{RGPS}}(\{\mathbf{R}_i, \mathbf{t}_i\}_{i=1}^{N_t}) = \sum_{i=1}^{N_t-1}\left\|(\mathbf{c}_i - \mathbf{c}_{i+1}) - \left(\mathbf{c}^{\text{init}}_i - \mathbf{c}^{\text{init}}_{i+1}\right)\right\|^2. \tag{20}$$

D. Single trace pose optimisation

For a single trace, we now optimise a hybrid objective function comprising a weighted sum of the data term and all priors. We initialise using the process described in Sec. VI-A and then optimise using nonlinear least squares. Specifically, we use the Levenberg-Marquardt algorithm as implemented in Ceres [76] which exploits Jacobian sparsity to improve efficiency and automatically computes analytical first order derivatives.

VII. Incremental multi-trace optimisation

An orthomosaic of a complete road network is likely to comprise many traces. Each trace may overlap with many others and to obtain a seamless orthomosaic, all overlapping regions must be optimised together. One solution would be to combine all traces into a single one and run the optimisation in the previous section. However, even with the simplifying assumptions made, this does not scale to millions of images. Instead, we propose an incremental process in which we optimise a single trace at a time but ensure that the new trace aligns with the previously optimised traces. This ensures that the size of any one optimisation problem is kept small.

Whenever a new trace is added into the network we use the fast search technique discussed in Sec. IV-B to find potential overlaps between the new trace and the existing traces and obtain filtered matches between them. Then, for each previously optimised trace s that overlaps with the new trace t, we add an additional cost:

$$\varepsilon_{\text{multi}}(\{\mathbf{R}_{t,j}, \mathbf{t}_{t,j}\}_{i=1}^{N_t}) = \sum_{(f,g)\in\mathcal{M}_{i,j}^{s,t}} \left\| h\left(\mathbf{H}_{t,j}^{-1}\begin{bmatrix}\mathbf{x}_{t,j,g}\\ 1\end{bmatrix}\right) - h\left(\mathbf{H}_{s,i}^{-1}\begin{bmatrix}\mathbf{x}_{s,i,f}\\ 1\end{bmatrix}\right)\right\|^2, \tag{21}$$

where $\mathbf{x}_{t,j,g}$ refers to the gth feature in image j in trace t and $\mathbf{H}_{t,j}$ is the homography from ground plane to the jth image in trace t, i.e. that is constructed from $\mathbf{R}_{t,j}$ and $\mathbf{t}_{t,j}$ using (7).

The overall optimisation for trace t comprises the single trace objective as in Sec. VI-D with additional objectives of the form (21), one for each overlapping trace. This additional cost will minimise the misalignment between the current and existing optimised traces.

VIII. Tile-based Image Stitching

This section describes our strategy for stitching images into a set of tiles that together form a seamless orthomosaic. There are two challenges in doing so. First, a stitched tile image must combine information from all of the images that overlap the tile without introducing artefacts such as seams at projected image boundaries or blurring caused by imprecise alignment. Second, images for adjacent tiles must appear seamless when placed side by side. To remain computationally tractable we cannot stitch all tiles simultaneously and so we must impose constraints between adjacent tiles whilst allowing tiles to be stitched in any order. Our approach is based on gradient domain blending [48] with two novel modifications: 1. gradients are selected from the source image pixel with maximum resolution and, 2. the introduction of boundary constraints when stitching a tile where one or more neighbours have already been stitched.

A. Per-pixel blending weights

Since our camera provides an oblique perspective view of the road surface, the camera-road distance (and hence the projected resolution) varies with pixel position. Specifically, pixels towards the bottom of the image are closer to the camera and are observed at higher resolution. We use this principle to compute a weight for each pixel in each captured image and project these weights to tile pixels. These are subsequently used as the blending weights for gradient domain tile stitching.

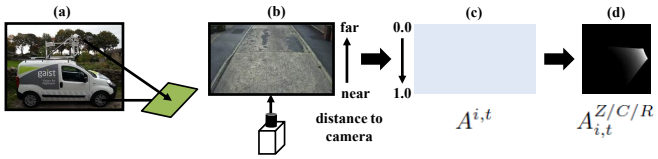

Fig. 5. Creating and projecting a weight image: (a) the camera mounted on the roof of the vehicle tilted towards the road surface, (b) the bottom part of the image closer to the camera is observed at higher resolution, (c) the weight coefficient of each pixel is inversely proportional to the distance from the camera, (d) projection of (c) to a tile.

Specifically, for image i in trace t, we compute a weight image $A^{i,t}(x,y) \in \mathbb{R}_{\geq 0}$. We consider two alternatives for these weight images. First, the exact camera-ground distance for each pixel could be computed using the estimated pose. This involves undistorting a pixel coordinate, projecting to the ground plane using the estimated pose via (9) then computing the distance between this point and the camera centre. However, this must be done for each image independently which is expensive. The second option uses a much simpler alternative. We linearly vary the weight according to the pixel row such that the top row has zero weight and the bottom has weight one, i.e. $A^{i,t}(x,y) = y/H^{\text{captured}}$. While not exact, this has the advantage of being very cheap to compute and constant across all images. We use this second strategy in our implementation.

Finally, for all images that overlap tile $Z/C/R$, i.e. where $(i,t) \in \mathcal{P}(Z/C/R)$, we apply the projection and resampling method described in Sec. III-C to project the weight image $A^{i,t}(x,y)$ into the tile, yielding $A_{i,t}^{Z/C/R}(X,Y)$. Tile pixels that are not covered by the projection are assigned a weight of 0. This process is illustrated in Fig. 5.

B. Seamless orthomosaics within a tile

We now perform gradient domain blending in order to combine all images that overlap a tile and obtain seamless orthomosaics within that tile. This amounts to solving a large, sparse system of linear equations in which the unknowns are the image intensities for the stitched tile, i.e. $I_c^{Z/C/R}(X,Y)$. The reconstructed intensities seek to preserve the image gradients selected from the image with highest weight, i.e. from the image that observed that point with highest resolution.

Specifically, the target gradients are approximated using forward finite difference and computed as:

$$G_x^c(X,Y) = I_{c,i^*,j^*}^{Z/C/R}(X+1,Y) - I_{c,i^*,j^*}^{Z/C/R}(X,Y), \tag{22}$$

$$G_y^c(X,Y) = I_{c,i^*,j^*}^{Z/C/R}(X,Y+1) - I_{c,i^*,j^*}^{Z/C/R}(X,Y), \tag{23}$$

where (i^*,t^*) is the image with highest weight for that pixel:

$$(i^*,t^*) = \underset{(i,t)\in\mathcal{P}(Z/C/R)}{\arg\max} A_{i,t}^{Z/C/R}(X,Y), \tag{24}$$

and $I_{c,i,j}^{Z/C/R}$ is the ground plane projection of colour channel c of image i in trace t computed as described in Sec. III-C.

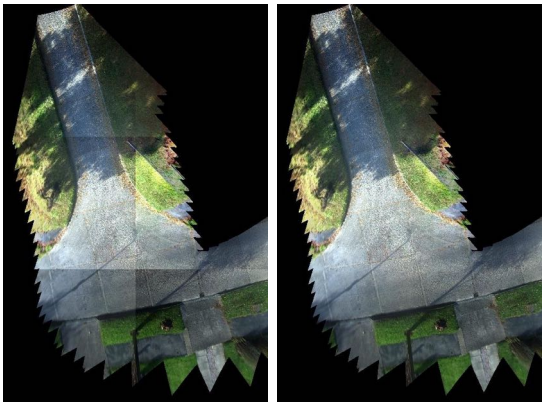

Fig. 6. Left: Visible seams at the boundaries of tiles. Right: Smooth colour transition between tiles yet still computed independently.

We switch to backward finite difference at the right hand boundary of the image. An image can only be reconstructed from gradients up to an unknown offset. To resolve this indeterminacy and to encourage the overall brightness and colour to match the original captured images, we include guide intensities as regularisation. This is simply computed by averaging all the ground plane images (ignoring pixels with zero weight). We denote this average image as $I^c_{\text{guide}}(X, Y)$.

The intensities for a stitched tile in colour channel c can now be computed by solving the following system of linear equations in a least squares sense:

$$\begin{bmatrix} \mathbf{D}_X \\ \mathbf{D}_Y \\ \lambda \mathbf{I} \end{bmatrix} \text{vec}(I_c^{Z/C/R}) = \begin{bmatrix} \text{vec}(G_x^c) \\ \text{vec}(G_y^c) \\ \lambda \text{vec}(I^c_{\text{guide}}) \end{bmatrix}. \quad (25)$$

Here, the matrices $\mathbf{D}_X, \mathbf{D}_Y \in \mathbb{R}^{n^2 \times n^2}$ compute finite difference approximations of the gradient in the X and Y directions respectively with $n = H^{\text{tile}} = W^{\text{tile}}$. These matrices are sparse (only two non-zero entries per row using finite differences). $\mathbf{I}$ denotes the identity matrix and λ is the regularisation weight which can be set to a small value. The solution to this system of equations can be seen as recovering detail from the gradient in the best image but overall brightness from the average of all images. Such systems can be solved efficiently meaning the tile stitching process could potentially be done in real time.

C. Seamless orthomosaics between tiles

If an orthomosaic tile is stitched independently using the process in Sec. VIII-B, then boundaries between adjacent tiles will be visible, as illustrated in Fig. 6 (left side). This problem is particularly noticeable when camera exposure vary dramatically between images. We resolve this problem by including overlaps with previously stitched neighbouring tiles.

This is done by extending the size of the guide image, I^c_{guide}, for e pixels equally at each side to create a small overlap with the neighbouring tiles. We solve for the unknown intensities over this larger $m \times m$ image but only include gradient constraints within the original tile region. Hence, $\mathbf{D}_x$ and $\mathbf{D}_y$ are modified to have size $n^2 \times m^2$ and the target gradient images retain their size of $n \times n$. We replace the fixed λ with a per-pixel weight encoded as a vector $\boldsymbol{\lambda} \in [0, 1]^{m^2}$. If the ith vectorised pixel lies within the overlap region we have only the guide intensity constraint and so $\lambda_i = 1$. If an adjacent tile has not yet been stitched, we set $\lambda_i = 0$ and I^c_{guide} to an arbitrary value for those pixels in the overlap region.

Since the guide intensities within the interior of the tile are formed by averaging the overlapping images, they may be inconsistent with the guide intensities in the overlap regions that are provided by previously stitched tiles. For this reason, we set $\lambda_i = 0$ if the ith vectorised pixel lies on the boundary of the tile interior. For interior pixels not lying on the boundary, we increase λ_i linearly over a distance e from the boundary with the remainder set to a constant, for which we use $\lambda_i = 0.1$. The modified system of equations is:

$$\begin{bmatrix} \mathbf{D}_X \\ \mathbf{D}_Y \\ \text{diag}(\boldsymbol{\lambda}) \end{bmatrix} \text{vec}(I_c^{Z/C/R}) = \begin{bmatrix} \text{vec}(G_x^c) \\ \text{vec}(G_y^c) \\ \text{diag}(\boldsymbol{\lambda}) \text{vec}(I^c_{\text{guide}}) \end{bmatrix}. \quad (26)$$

The guide intensities in the overlap regions cause tile boundaries to exactly match with adjacent tiles. Any large intensity changes that would have occurred are made less visible as low frequency transitions. This allows us to stitch tiles one at a time but still produce a seamless mosaic (see Fig. 6, right).

Although the system is sparse and thus quick to solve, we can still improve speed by reducing the number of equations for guide intensity constraints. Instead of including all the pixels in the interior, we can only include a reduced subsample. Not only does this improve speed, the quality of the stitched image is also improved because it avoids the solution being encouraged towards the overly smooth average image.

D. Orthomosaic map generation

Of course, the use of previously stitched tiles as constraints when stitching a new tile means that the overall result is dependent on the order in which tiles are stitched. In addition, if stitching is parallelised then care must be taken to avoid independently stitching neighbouring tiles. To avoid this, we feed tiles to the concurrently running threads such that they have R or C different by at least 2, i.e. are non-adjacent. Finally, an orthomosaic for any region can be constructed by simply placing the tiles side by side according to its tile positions. In practice, a user interacts with the map by changing the region of interest (e.g. by dragging the map) or the zoom level, which generates requests for a set of tiles. These are either returned directly (if they were precomputed in advance) or stitched on-the-fly in an on-demand implementation. In principle, the entire orthomosaic could be created by concatenating all stitched tiles together into one large map.

IX. Experiments

Data The datasets presented in this study were collected by seven vehicles equipped with a downward facing camera and a GPS receiver, both of which were temporally synchronised. The settings of these cameras, e.g. the downward angle and the height of the camera on the rig, were roughly hand measured whilst fitting, for use as initial estimates for pose optimisation, and the cameras are all pre-calibrated.

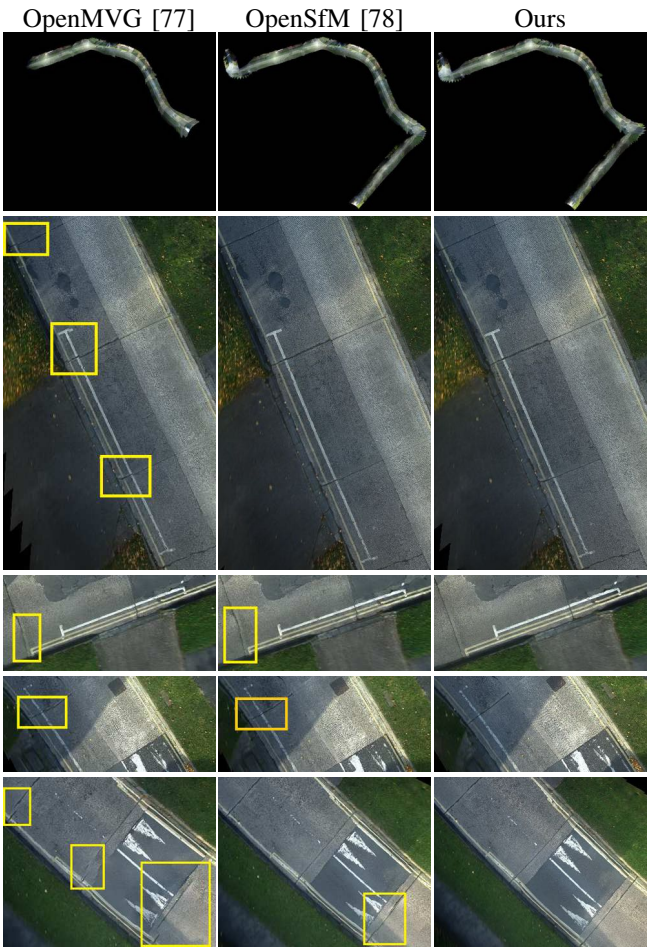

Fig. 7. Qualitative comparison on a 400-image sequence with a simple trajectory. Orthomosaics are created using the estimated camera poses from Left: OpenMVG [77], Middle: OpenSfM [78] and Right: ours. Our work has shown to be superior to the two approaches, noticeably at the crossing lines highlighted as yellow boxes. (Zoom for detail).

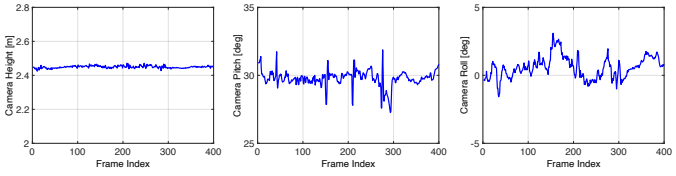

Fig. 8. Camera pose obtained on a 400-image trajectory. Left: heights remain close to the approximate height of the camera above the ground, Middle: pitch fluctuates when the vehicle goes over uneven surfaces, Right: rolls deviate from 0° when the vehicle is cornering.

Pose estimation evaluation We first evaluated the reliability of our proposed motion-from-homographies by comparing against two[4] state-of-the-art SfM implementations: OpenMVG [77] (based on global SfM) and OpenSfM [78] (incremental SfM). Fig. 7 compares the orthomosaics created using the poses estimated by these two approaches and ours on a 400-street-level-image sequence with GPS. OpenMVG was unable to obtain poses at the start and end of the trajectory and the inaccurate estimated poses at the end caused the road to appear

[4]Note that other potential comparison methods, e.g. COLMAP [22], VisualSfM [23], Bundler [26] and MVE [79], do not support GPS priors, hence give much worse performance and so the comparison is unfair.

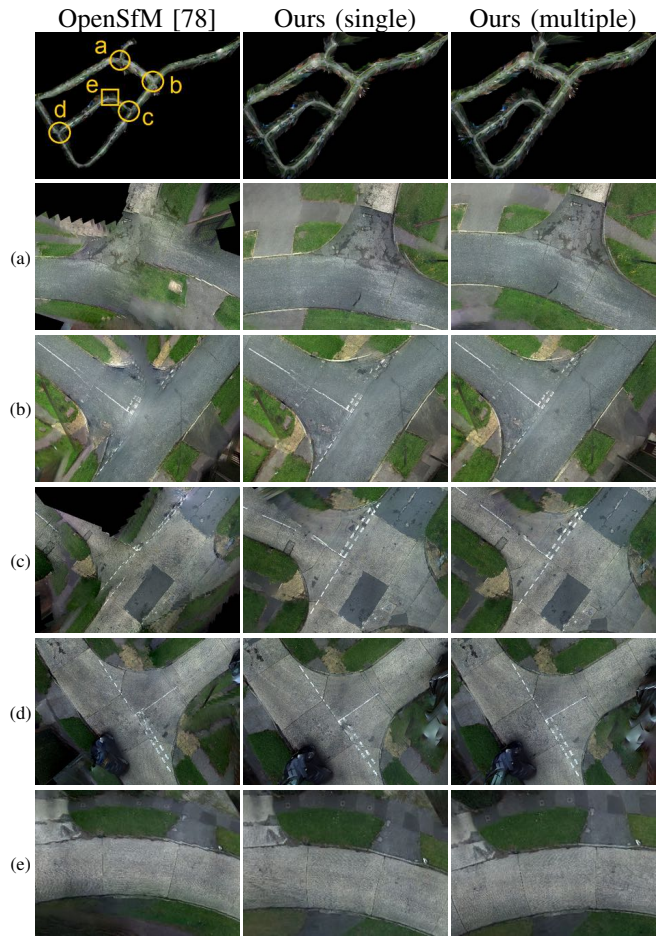

Fig. 9. Qualitative results on a crossing and overlapping two-trace dataset. Left: OpenSfM [78], Middle: Our method with single-trace optimisation and Right: Our method with multiple-trace optimisation. First row: orthomosaics of the entire scene. Other rows, corresponding to the points (a)-(e) labelled on the top-left image, show that our methods produce superior results especially when two traces crossing each other. (Zoom for detail).

wider. Our work has shown to be superior to OpenMVG in most parts of the orthomosaics and achieved slightly better results than OpenSfM in some areas, see the bottom three rows of Fig. 7. With 3D structures being eliminated, our approach consumed only 21 sec for bundle adjustment (reduced to 9 sec when running with multiple threads) while [77] and [78] took 17 mins and 2.47 hours, respectively. We excluded the time taken for feature extraction and matching.

We quantitatively evaluate pose estimation by measuring the error between observed and predicted groundpoints, using (9). We labelled 7 groundpoints and 30 corresponding image GCPs. Note that these GCPs were used only for evaluation, not for pose optimisation. The errors were 0.27 ± 0.095m (ours), 0.27 ± 0.094m (OpenMVG) and 1.61 ± 0.14m (OpenSfM). Hence, our performance matches OpenMVG while requiring significantly less computation time and noting that OpenMVG fails to compute poses for about half of the sequence.

Our estimated camera poses correspond well with our objective functions. Fig. 8, left shows the height of the camera remains approximately constant and close to the measured height. Similarly, pitch angles (middle) remain almost constant at around 30°, and both heights and pitches change

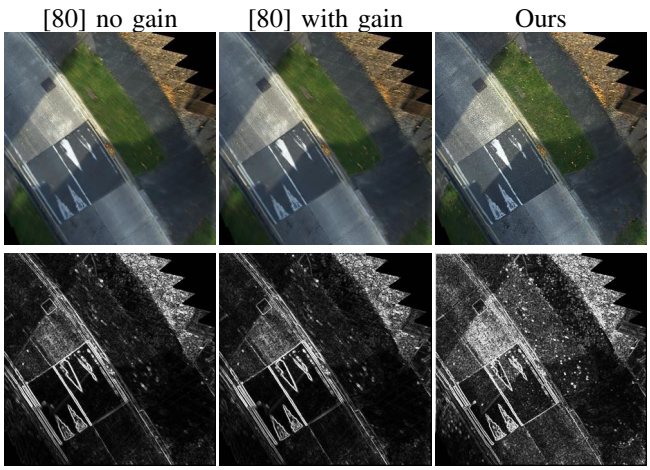

Fig. 10. Qualitative comparison of image blending from Left: [80] no gain compensation, Middle: [80] with gain compensation and Right: Our proposed work. Top row: stitching result, bottom row: gradient magnitude.

significantly only when the vehicle goes over uneven surfaces e.g. speed bumps. The roll angles (right) are close to 0° most of the time except when cornering during frames 150-170.

We also evaluated our method in comparison with OpenSfM [78] on a two-trace dataset (790-image and 484-image sequences) crossing and overlapping each other. We tested our motion-from-homographies on two modes. For the first mode, similar to that given to OpenSfM, these two traces were combined and optimised as a single trace. For the second mode, which is the typical way we deal with multiple traces, we started by optimising one trace then incrementally optimised the second trace based on the multiple-trace optimisation. Our method consumed 11.26 mins using the single-trace based approach and only 2.35 mins with the multiple-trace approach for bundle adjustment, while OpenSfM consumed 53 hours. Our proposed work produced superior alignments where the two traces overlap while OpenSfM failed to connect two parts of the road, presented as (a)-(d) in Fig. 9, and produced misalignment artefacts, the bottom row of Fig. 9.

Image-stitching evaluation We compared our image stitching against [80]. Since we wish to produce a virtual top down view of the stitched images, we used our camera poses computed from the previous experiment for image registration then used [80] to stitch the images via multi-band blending with and without gain compensation, see Fig. 10 for comparison.

To quantitatively evaluate how well our stitching method can preserve fine features in the produced results, we computed the gradient magnitude averaged over foreground pixels as a measure of sharpness in a stitched image. Averaging over images, [80] without gain compensation yielded a mean gradient magnitude of 0.128, with gain compensation this was 0.126, while our method produced the sharpest images with 0.218. We conclude that our stitching approach can preserve fine details significantly better than [80].

Comparison with state-of-the-art We wish to compare our proposed system against many state-of-the-art systems. Due to the requirements of the captured images to be orthogonal to the ground [64], [63], lack of orthomosaics features in AutoStitch[5], OpenPano[6], issues with memory management [65], we could only be able to compare our system against Agisoft Photoscan[7](AS) and Context Capture[8](CC). We use the same dataset as that used in **Pose estimation evaluation**, and the computational results are presented in Tab. I. Ours pose estimation consumes one third processing time of that used for AP but without the need for computing mesh and is significantly less than that of CC. The stitching time used in ours is twice that of the two. This is because our work generate each tile independently and that no data are shared between them. However, since minimum computational resource (particularly memory) is used to create a tile, this enables us to generate several tiles in parallel up to the limit of the processor (in this experiment, we ran 8 threads in parallel). Overall, ours consumes less processing time than these two commercial software. Fig. 12 compares the orthomosaics generated by AP, CC, MGRAPH[9] and ours. Our results are competitive with AP and CC showing no artefacts between the stitched images, while discontinuity seams around image boundaries are visible in MGRAPH results as their orthomosaics are generated without blending and even look more blurry. AP and ours have shown to preserve fine details where CC, though impressive, losses some sharpness and washes out some detail in some parts (this is likely because their system also eliminates shadow presenting in the stitched images). We conclude that orthomosaics generated from our novel system is comparable as that from commercial products.

TABLE I
COMPARISON WITH STATE-OF-THE-ART

	Mosaic-size[a] (pix)	time (pose)	time (mesh)	time (stitch)
AP	21776×17807	30m	51m	11m
CC	22600×16384	2h 40m	–	10m
Ours	19968×16896	10m	–	22m

[a] The difference in orthomosaics size is due to the background pixels padded to the final results.

Large-scale evaluation To demonstrate the performance of our proposed methods on large-scale datasets, we used a dataset of 1.4 million images, made up from 1,700 traces acquired by three vehicles driving around the city of York, UK, for a period of ten days during summer daytime. The dataset covered roughly 350km^2, measured from the extreme boundary. We processed the whole dataset on a 12-core CPU with a 128-GB RAM. We relied on parallel processing, in which we extracted features and obtained matches for self-overlaps, as discussed in Sec. IV-A, for individual traces in parallel. Then we started by running our motion-from-homographies on the traces which were provided with GCPs or GCLs to obtain camera poses and treated them as the established trace network. After that we extended the road

[5]http://matthewalunbrown.com/autostitch/autostitch.html
[6]https://github.com/ppwwyyxx/OpenPano
[7]https://www.agisoft.com
[8]https://www.bentley.com/en/products/brands/contextcapture
[9]Since MGRAPH does not work with oblique images, we used the camera poses computed by our system for image registration and used their approach [63] to create orthomosaics.

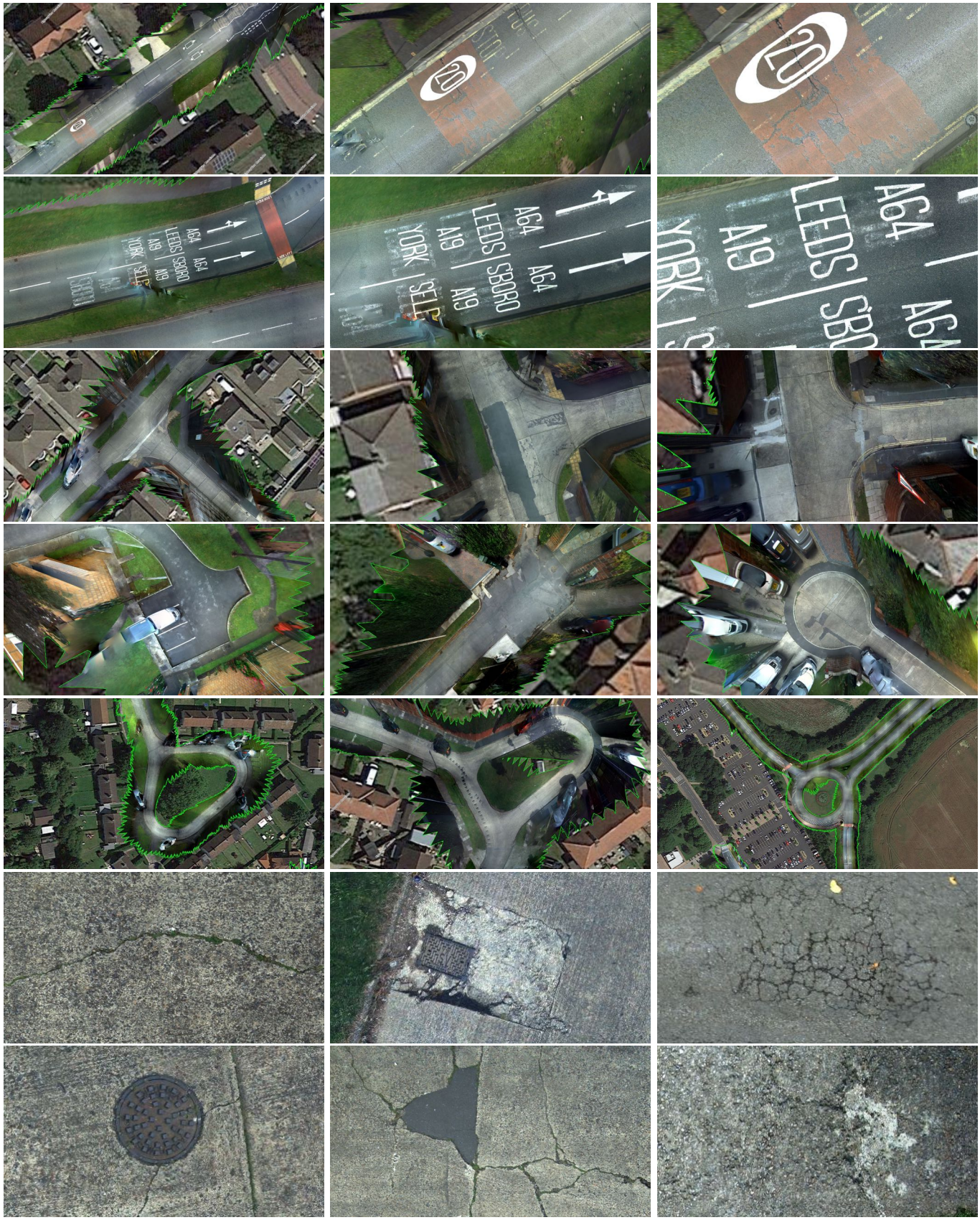

Fig. 11. Qualitative results on a 1.4 million-image dataset. Orthomosaics are generated and overlaid on top of aerial images (boundary between orthomosaic and aerial image shown in green). First row: stitched images from a single trace. Second row: stitched images from two traces running in opposite direction. Third row: junctions where at least two traces crossing and joining each other. Fourth row: no through road where the vehicle had to re-trace or made a u-turn. Fifth row: loops or roundabout. Two bottom rows: fine details of road defects extracted from our orthomosaics. (Zoom for detail).

Fig. 12. Qualitative comparison of orthomosaics generated (from left to right) by Agisoft Photoscan, Context Capture, MGRAPH[9] and ours. (Zoom for detail).

network by incrementally optimising traces that overlapped with any of the established traces, Sec. IV-B, and this step was run in parallel. Once completed, we stitched each tile in parallel, and the final orthomosaics were simply created by placing tiles side by side based on its geo-positions. We generated in total 40.5 billion tiles for 4 zoom levels: $Z = 21, \ldots, 24$, equivalent to a 10^{15} (i.e. petapixel) image. The most time consuming process lied in feature matching and filtering. Processing the entire dataset took approximately 4 weeks, though this can be reduced to 2 weeks using GPU-based implementations of feature matching and RANSAC.

The results are presented in Fig. 11 and a video visualisation is available[1]. The top two rows illustrate parts of the road network from a large coverage scale down to finer details. The first row presents an orthomosaic made up of a single trace while the results on the second row comprise at least two traces running in the opposite direction. The third row shows the results at junctions where images from at least two traces were incrementally optimised. The fourth and fifth rows demonstrate the results at no through road and loops or roundabout where the vehicle capturing images had to retrace itself or made a u-turn and where multiple traces overlap each other. The two bottom rows illustrate orthomosaics at the scale where defects on road surface are clearly visible. The orthomosaics presented here are composed of tiles, which were stitched independently, showing no seam artefacts at the boundaries between tiles.

X. Conclusion and Discussion

In this paper, we have presented a complete pipeline to build city-scale orthomosaics of the road network from a scale where the entire network is visible down to a scale where fine details

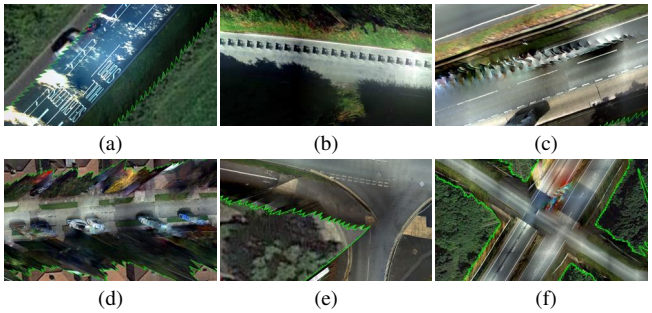

Fig. 13. Failure cases. Artefacts in orthomosaics due to (a) scene shadow, (b) observer shadow, (c) moving objects, (d) stationary objects, (e) out of sight, (f) mixing of multiple mosaics of different elevations. (Zoom for detail).

on road surfaces can be inspected. We believe the work makes two important contributions.

First, we proposed motion-from-homographies to estimate camera poses of street-level images as an alternative to SfM. With the assumption that the road surface is planar and induces no 3D structure, motion-from-homographies reduces the complexity of the optimisation problem by limiting to solve only 6 unknowns for each camera and completely eliminating the reconstruction of 3D scene points. Our approach dramatically reduces the time to solve for camera poses, while still producing accurate alignment compared with conventional SfM techniques [77], [78]. Second, we presented a tile-based image stitching that can create very large orthomosaics from millions of images and yet still produce a visually pleasing result. The stitching is operated on tiles allowing us to be able to stitch very large datasets and enables the large-scale orthomosaics to be viewed online. Our gradient domain stitching ensures the tiles preserve maximum detail of the input images and seams are invisible both within the tiles and between tiles.

There are still some areas for improvement and we show failure cases in Fig. 13. First, street-level images inevitably include shadows which cause artefacts when they change, Fig. 13(a) and (b). Shadow detection could be used to remove shadow pixels from the stitching process. For observer shadows, we could use the geometrical model of the observer to predict shadow masks [81]. Second, moving objects such as cars create ghosting artefacts in the stitched mosaics, Fig. 13(c).

We could use semantic segmentation or object detection [82] to learn to detect undesirable objects such as parked cars, Fig. 13(d), and exclude them in the stitching process. Third, artefacts occur when part of a junction is out-of-sight when a vehicle makes a turn, Fig. 13(e), which can result in stitching failure when multiple traces overlap. Finally, our approach does not account for two or more roads sharing the same 2D geo-positions, e.g. where a bridge crosses another road. In this case, their contents were blended together in the same tiles, Fig. 13(f). Solving this would require estimation of altitude to detect when two overlapping traces are not the same road, then stitched and visualised as different map layers.

Acknowledgment

This work was supported by Innovate UK grant KTP 9627. W. Smith was supported by a Royal Academy of Engi-

neering/The Leverhulme Trust Senior Research Fellowship. We thank Paulina Lewinska for help obtaining results for comparison methods.

REFERENCES

[1] B. Klingner, D. Martin, and J. Roseborough, "Street view motion-from-structure-from-motion," in *Proc. ICCV*, 2013, pp. 953–960.

[2] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.

[3] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, Nara, Japan, November 2007.

[4] H. Lim, J. Lim, and H. J. Kim, "Real-time 6-dof monocular visual slam in a large-scale environment," in *ICRA*, 2014, pp. 1532–1539.

[5] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "Orb-slam: A versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[6] M. Yokozuka, S. Oishi, S. Thompson, and A. Banno, "Vitamin-e: Visual tracking and mapping with extremely dense feature points," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, jun 2019, pp. 9633–9642. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CVPR.2019.00987

[7] J. Engel, J. Stückler, and D. Cremers, "Large-scale direct slam with stereo cameras," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 1935–1942.

[8] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[9] R. Wang, M. Schwörer, and D. Cremers, "Stereo DSO: large-scale direct sparse visual odometry with stereo cameras," in *ICCV 2017*. IEEE Computer Society, 2017, pp. 3923–3931. [Online]. Available: https://doi.org/10.1109/ICCV.2017.421

[10] T. Schops, T. Sattler, and M. Pollefeys, "Bad slam: Bundle adjusted direct rgb-d slam," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[11] L. Zhao, Z. Liu, J. Chen, W. Cai, W. Wang, and L. Zeng, "A compatible framework for rgb-d slam in dynamic scenes," *IEEE Access*, vol. 7, pp. 75 604–75 614, 2019.

[12] S. Leutenegger, P. Furgale, V. Rabaud, M. Chli, K. Konolige, and R. Siegwart, "Keyframe-based visual-inertial slam using nonlinear optimization," 06 2013.

[13] R. Mur-Artal and J. D. Tardós, "Visual-inertial monocular slam with map reuse," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 796–803, 2017.

[14] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[15] P. Geneva, K. Eckenhoff, and G. Huang, "A linear-complexity ekf for visual-inertial navigation with loop closures," in *ICRA*, 2019, pp. 3535–3541.

[16] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *ECCV*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 834–849.

[17] G. Cioffi and D. Scaramuzza, "Tightly-coupled fusion of global positional measurements in optimization-based visual-inertial odometry," in *IROS*, 2020.

[18] H. Strasdat, J. M. M. Montiel, and A. J. Davison, "Scale drift-aware large scale monocular slam," in *In Proceedings of Robotics: Science and Systems*, 2010.

[19] G. Bresson, Z. Alsayed, L. Yu, and S. Glaser, "Simultaneous localization and mapping: A survey of current trends in autonomous driving," *IEEE T-IVT*, vol. 2, no. 3, pp. 194–220, 2017.

[20] G. Huang, "Visual-inertial navigation: A concise review," in *ICRA*, 2019, pp. 9572–9582.

[21] B. Huang, J. Zhao, and J. Liu, "A survey of simultaneous localization and mapping with an envision in 6g wireless networks," 2020.

[22] J. L. Schönberger and J.-M. Frahm, "Structure-from-motion revisited," in *Proc. CVPR*, 2016.

[23] C. Wu, "Towards linear-time incremental structure from motion," in *Proc. 3DV*, June 2013, pp. 127–134.

[24] N. Snavely, S. M. Seitz, and R. Szeliski, "Photo tourism: Exploring photo collections in 3d," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 835–846, 2006.

[25] C. Sweeney, V. Fragoso, T. Höllerer, and M. Turk, "Large scale sfm with the distributed camera model," in *Proc. 3DV*, 2016, pp. 230–238.

[26] N. Snavely, S. M. Seitz, and R. Szeliski, "Modeling the world from internet photo collections," *Int. J. Comput. Vision*, vol. 80, no. 2, pp. 189–210, 2008.

[27] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski, "Building rome in a day," in *Proc. ICCV*, 2009, pp. 72–79.

[28] J. Heinly, J. L. Schönberger, E. Dunn, and J. M. Frahm, "Reconstructing the world* in six days," in *Proc. CVPR*, June 2015, pp. 3287–3295.

[29] K. Cornelis, F. Verbiest, and L. V. Gool, "Drift detection and removal for sequential structure from motion algorithms," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 10, pp. 1249–1259, 2004.

[30] Z. Cui and P. Tan, "Global structure-from-motion by similarity averaging," in *Proc. ICCV*, Dec 2015, pp. 864–872.

[31] C. Sweeney, T. Sattler, T. Höllerer, M. Turk, and M. Pollefeys, "Optimizing the viewing graph for structure-from-motion," in *Proc. ICCV*, Dec 2015, pp. 801–809.

[32] P. Moulon, P. Monasse, and R. Marlet, "Global fusion of relative motions for robust, accurate and scalable structure from motion," in *Proc. ICCV*, December 2013.

[33] V. M. Govindu, "Lie-algebraic averaging for globally consistent motion estimation," in *Proc. CVPR*, 2004, pp. 684–691.

[34] R. Hartley, K. Aftab, and J. Trumpf, "L1 rotation averaging using the weiszfeld algorithm," in *Proc. CVPR*, June 2011, pp. 3041–3048.

[35] A. Chatterjee and V. M. Govindu, "Efficient and robust large-scale rotation averaging," in *Proc. ICCV*, Dec 2013, pp. 521–528.

[36] P. Moulon, P. Monasse, and R. Marlet, "Global fusion of relative motions for robust, accurate and scalable structure from motion," in *Proc. ICCV*, Dec 2013, pp. 3248–3255.

[37] K. Wilson and N. Snavely, "Robust global translations with 1dsfm," in *Proc. ECCV*, vol. 8691, 2014, pp. 61–75.

[38] S. N. Sinha, D. Steedly, and R. Szeliski, "A multi-stage linear approach to structure from motion," in *Proc. ECCV*, 2012, pp. 267–281.

[39] D. Crandall, A. Owens, N. Snavely, and D. Huttenlocher, "Discrete-continuous optimization for large-scale structure from motion," in *Proc. CVPR*, 2011, pp. 3001–3008.

[40] R. Carceroni, A. Kumar, and K. Daniilidis, "Structure from motion with known camera positions," in *Proc. CVPR*, 2006, pp. 477–484.

[41] M. Lhuillier, "Incremental fusion of structure-from-motion and gps using constrained bundle adjustments," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 12, pp. 2489–2495, Dec 2012.

[42] H. AliAkbarpour, K. Palaniappan, and G. Seetharaman, "Fast structure from motion for sequential and wide area motion imagery," in *Proc. ICCV-W*, Dec 2015, pp. 1086–1093.

[43] A. L. Rodríguez, P. E. L. de Teruel, and A. Ruiz, "Reduced epipolar cost for accelerated incremental SfM," in *Proc. CVPR*, 2011, pp. 3097–3104.

[44] V. Indelman, R. Roberts, C. Beall, and F. Dellaert, "Incremental light bundle adjustment," in *Proc. BMVC*, 2012, pp. 134.1–134.11.

[45] Z. Zhou, H. Jin, and Y. Ma, "Robust plane-based structure from motion," in *Proc. CVPR*, June 2012, pp. 1482–1489.

[46] S. Lovegrove, A. J. Davison, and J. Ibanez-Guzmán, "Accurate visual odometry from a rear parking camera," in *Proc. IV*, 2011, pp. 788–793.

[47] H. S. Sawhney, S. Hsu, and R. Kumar, "Robust video mosaicing through topology inference and local to global alignment," in *Proc. ECCV*. Springer, 1998, pp. 103–119.

[48] P. Pérez, M. Gangnet, and A. Blake, "Poisson image editing," *ACM Trans. Graph.*, vol. 22, no. 3, pp. 313–318, 2003.

[49] A. Levin, A. Zomet, S. Peleg, and Y. Weiss, "Seamless image stitching in the gradient domain," in *Proc. ECCV*, 2004, pp. 377–389.

[50] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick, "Graphcut textures: Image and video synthesis using graph cuts," *ACM Trans. Graph.*, vol. 22, no. 3, pp. 277–286, 2003.

[51] A. Agarwala, M. Dontcheva, M. Agrawala, S. Drucker, A. Colburn, B. Curless, D. Salesin, and M. Cohen, "Interactive digital photomontage," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 294–302, 2004.

[52] N. Gracias, M. Mahoor, S. Negahdaripour, and A. Gleason, "Fast image blending using watersheds and graph cuts," *Image Vis. Comput.*, vol. 27, no. 5, pp. 597–607, Apr. 2009.

[53] A. A. Efros and W. T. Freeman, "Image quilting for texture synthesis and transfer," in *Proc. SIGGRAPH*, 2001, pp. 341–346.

[54] Y. Xiong and K. Pulli, "Fast panorama stitching for high-quality panoramic images on mobile phones," *IEEE Trans. Consum. Electron.*, vol. 56, no. 2, pp. 298–306, 2010.

[55] H. Wu, S. Zheng, J. Zhang, and K. Huang, "GP-GAN: towards realistic high-resolution image blending," *CoRR*, vol. abs/1703.07195, 2017.
[56] Y.-H. Tsai, X. Shen, Z. Lin, K. Sunkavalli, and M.-H. Yang, "Sky is not the limit: Semantic-aware sky replacement," *ACM Trans. Graph.*, vol. 35, no. 4, pp. 149:1–149:11, Jul. 2016.
[57] A. Agarwala, "Efficient gradient-domain compositing using quadtrees," *ACM Trans. Graph.*, vol. 26, no. 3, p. 94, 2007.
[58] M. Kazhdan, D. Surendran, and H. Hoppe, "Distributed gradient-domain processing of planar and spherical images," *ACM Trans. Graphic.*, vol. 29, no. 2, p. 14, 2010.
[59] J. Kopf, M. Uyttendaele, O. Deussen, and M. F. Cohen, "Capturing and viewing gigapixel images," *ACM Trans. Graph.*, vol. 26, no. 3, Jul. 2007.
[60] R. Prados, R. Garcia, N. Gracias, J. Escartin, and L. Neumann, "A novel blending technique for underwater gigamosaicing," *IEEE J. Ocean. Eng.*, vol. 37, no. 4, pp. 626–644, Oct 2012.
[61] J. Ferrer, A. Elibol, O. Delaunoy, N. Gracias, and R. Garcia, "Large-area photomosaics using global alignment and navigation data," in *Proc. MTS/IEEE OCEANS*, 2007.
[62] S. Bu, Y. Zhao, G. Wan, and Z. Liu, "Map2dfusion: Real-time incremental uav image mosaicing based on monocular slam," in *Proc. IEEE/RSJ IROS*, 2016, pp. 4564–4571.
[63] J. J. Ruiz, F. Caballero, and L. Merino, "Mgraph: A multigraph homography method to generate incremental mosaics in real-time from uav swarms," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 2838–2845, 2018.
[64] A. Kern, M. Bobbe, Y. Khedar, and U. Bestmann, "Openrealm: Real-time mapping for unmanned aerial vehicles," in *Proc. ICUAS*, 2020.
[65] E. Garcia-Fidalgo, A. Ortiz, F. Bonnin-Pascual, and J. P. Company, "Fast image mosaicing using incremental bags of binary words," in *Proc. ICRA*, 2016, pp. 1174–1180.
[66] A. M. Pinto, H. Pinto, and A. C. Matos, "A mosaicking approach for visual mapping of large-scale environments," in *Proc. ICARSC*, 2016.
[67] J. Heikkilä and O. Silven, "A four-step camera calibration procedure with implicit image correction," in *Proc. CVPR*, 1997.
[68] Google. (2017) Google maps apis. [Online]. Available: https://developers.google.com/maps
[69] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. The MIT Press, 2009.
[70] J. L. Bentley, D. F. Stanat, and E. Williams, "The complexity of finding fixed-radius near neighbors," *Inform. Process. Lett.*, vol. 6, no. 6, pp. 209 – 212, 1977.
[71] E. Haines, *Point in polygon strategies*, 1994, vol. 994.
[72] A. L. F. Meister, "Generalia de genesi figurarum planarum et inde pendentibus earum affectionibus," *Nov. Com. Gött. (in Latin)*, vol. 1, p. 144, 1769.
[73] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comp. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.
[74] I. L. Dryden and K. V. Mardia, *Statistical Shape Analysis*. Chichester: Wiley, 1998.
[75] E. Kaplan and C. Hegarty, *Understanding GPS: principles and applications*. Artech house, 2005.
[76] S. Agarwal, K. Mierle, and Others, "Ceres solver," http://ceres-solver.org.
[77] P. Moulon, P. Monasse, R. Marlet, and Others, "Openmvg. an open multiple view geometry library." https://github.com/openMVG/openMVG.
[78] Mapillary, "Opensfm. open source structure from motion pipeline." https://github.com/mapillary/OpenSfM.
[79] S. Fuhrmann, F. Langguth, and M. Goesele, "Mve: A multi-view reconstruction environment," in *Proc. Eurographics Workshop on Graphics and Cultural Heritage*, 2014, pp. 11–18.
[80] M. Brown and D. G. Lowe, "Automatic panoramic image stitching using invariant features," *IJCV*, vol. 74, pp. 59–73, 2007.
[81] S. Tanathong, W. A. P. Smith, and S. Remde, "Eliminating the observer effect: Shadow removal in orthomosaics of the road network," in *Proc. ICCV-W*, Oct 2017, pp. 262–269.
[82] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask r-cnn," in *Proc. ICCV*, Oct 2017.

Supannee Tanathong obtained her Ph.D in geoinformatics engineering from the University of Seoul, Seoul, Korea and a M.Eng and B.Eng in computer engineering from King Mongkut's University of Technology Thonburi, Bangkok, Thailand. She is currently a Computer Vision Scientist at Gaist Solutions, U.K. Before that she was a KTP Associate between the University of York, U.K. and Gaist Solutions.

William Smith received the BSc degree in computer science, and the PhD degree in computer vision from the University of York, York, United Kingdom. He is currently a Reader with the Department of Computer Science, University of York, York, United Kingdom. He holds a Royal Academy of Engineering/The Leverhulme Trust Senior Research Fellowship. His research interests are in shape and appearance modelling, model-based supervision and physics-based vision. He has published more than 100 papers in international conferences and journals.

Stephen Remde received the M.Eng degree in Software Engineering and the Ph.D. degree in operations research from the University of Bradford, Bradford, U.K. He is currently the CTO of Gaist Solutions Limited who provide world class, award winning technology for the assessment, and long term deterioration modelling of Highways and other large scale infrastructure. At Gaist he oversees the research, hardware, and software development teams and consequently has a keen interest in computer vision.