Colourising Point Clouds using Independent Cameras

Pavel Vechersky, Mark Cox, Paulo Borges, Thomas Lowe

***Abstract*— We investigate the problem of colourising a point cloud using an arbitrary mobile mapping device and an independent camera. The loose coupling of sensors creates a number of challenges related to timing, point visibility and colour determination. We address each of these problems and demonstrate our resulting algorithm on data captured using sensor payloads mounted to hand-held, aerial and ground systems, illustrating the 'plug-and-play' portability of the method.**

I. Introduction

In the last decade we have seen a dramatic increase in the development of mobile mapping systems, where a 3D geometric representation of the environment is generated with high accuracy. In this space, lidar-based systems are very popular, finding application in robotics (vehicle navigation), gaming (virtual reality), surveying, among other industries. Key advantages of lidar sensing over its main competitor (camera) are the invariance to lighting conditions and high-precision range information, making lidars an excellent and proven alternative for 3D mapping. On the other hand, a fundamental drawback of lidars compared to cameras is that they do not provide rich visual appearance information. Depending on the application, this type of information is of great benefit for human (and often machine) interpretation. An obvious solution is to fuse data from lidar with camera data, hence combining range with colour information. There are a number of strategies to perform this fusion, and some are tightly dependent on the devices and sensor setup. Our goal in this paper is to create a practical solution that is generic, and that can be applied to existing camera-less platforms, by simply adding a camera to any existing lidar 3D mapping device.

Consider, for example, the 3D mapping devices shown in Figure 1, to which we have added a camera. In this figure, the mapping strategies range from having the mapping sensor as hand-held, to mounted on aerial and ground platforms. The ability to easily add colour to point clouds generated by such different platforms has a number of advantages. They include: (*i*) economic attractiveness, as existing camera-less devices can be fitted with a camera, (*ii*) there is no restriction on the camera type or modality provided that it is of sufficient quality to generate reliable optical flow, (*iii*) many modern mapping devices are designed to be mobile, permitting increased colour accuracy from multiple candidate colours per point, (*iv*) portability and platform independence. These benefits serve as motivation for the method we propose in this paper.

The authors are with the Robotics and Autonomous System Group, Data61, CSIRO, Australia `first.last@csiro.au`

In general, the fundamental process to achieve colourised point clouds is to project the 2D camera images over the 3D points, such that colours (or other information such as hyperspectral data) are assigned to each 3D point. In this case, there are key fundamental challenges associated with colourising point clouds, which are (*i*) clock synchronisation between the lidar and the imaging device, (*ii*) determining the visibility of points and (*iii*) intelligently determining the colour assignments for each 3D point. To solve these challenges, our basic assumption is that the camera-less 3D mapping device outputs a point cloud (i.e., the 3D map) and the associated device's trajectory $\mathbf{t}$, as is the case in modern simultaneous localisation and mapping (SLAM) algorithms [1] [2] [3] [4]. In this context, we propose the following solutions:

A. Challenge 1: Clock Synchronisation

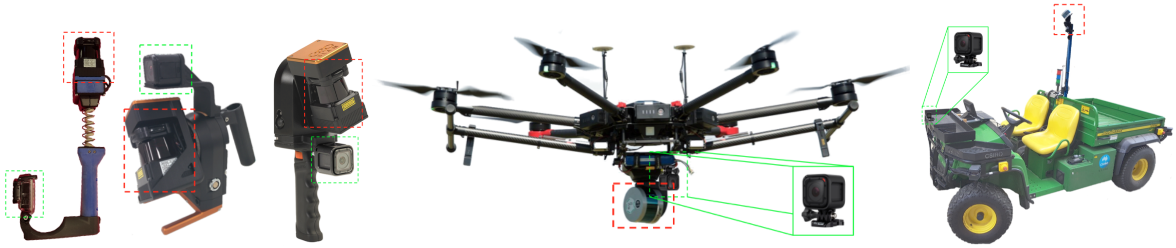
The camera, device and other modalities need to share a common clock before any processing can be performed.

To achieve synchronisation, we assume that the computed trajectory $\mathbf{t}$ can be transformed to a camera trajectory using a fixed rigid transformation. In Section III-A we outline an algorithm for temporally synchronising captured camera video with the computed device trajectory using the yaw-rate of the device and an estimate of yaw-rate computed from the camera video. The yaw-rate of the device can be computed from the device trajectory itself or using the output from an inertial measurement unit, if available. We then cross-correlate the yaw-rates from the camera and mapping device in order to obtain the temporal offset and scaling which relate the two modalities.

B. Challenge 2: Visibility of Points

Determining which points are visible from a specific viewpoint is crucial considering that the camera and the lidar can be mounted far from each other. The example shown in Figure 1e illustrates this scenario, where the camera is attached to the front of the vehicle while the lidar is on top.

To solve the visibility issue, we use the algorithm proposed in the seminal works of Katz et al. [8], [9], which does not require estimating surface normals nor estimates of point volumes. The algorithm applies a spherical reflection kernel to each point such that points closest to the camera are reflected to points that are furthest from the camera. A point is classified as visible if its reflection is a member of the convex hull of the reflected point cloud. The choice of kernel and associated parameters affect the resulting visibility determination. In Section IV we present a theorem which shows that the linear kernel proposed in [8], [9] is not scale



(a) (b) (c) (d) (e)

Fig. 1: Examples of mobile mapping devices with various lidar configurations, ranging from handheld (a-c), to aerial (d) and ground (e) platforms. The original lidar device is highlighted in red, while the added camera (which is not connected or synchronised to the lidar) is indicated by the green box. References for the lidar mapping devices: (a) Zebedee [2], (b) CSIRO Revo, (c) ZebCam [5], (d) Hovermap [6], (e) Gator [7].

Fig. 2: Example illustrating the "transparency" effect in 3D lidar point clouds. This transparency can cause points that are occluded in the real world to be erroneously coloured when projecting an image on the points.

invariant i.e. the scale of an object in the point cloud has an effect on its visibility. We show that the exponential kernel (also proposed in [9]) is scale invariant and achieves consistent results as a consequence, finding application in our colourisation problem.

C. Challenge 3: Colour Assignment

Once visibility is determined, the next challenge is how to assign a colour to a 3D point from a set of colour candidates. Considering mobile mapping (i.e., from a moving platform), the problem lies in the fact that a reconstructed 3D point is likely to have been seen in multiple video frames during data collection, and the appearance of that point varies throughout the acquisition process depending on the observation angle. This problem is much more severe in continuous mobile mapping than in traditional static "tripod" systems. Hence, we propose a robust rolling average colour scheme that can process colour observations sequentially, increasing the processing speed.

After addressing challenges 1-3 above, we present experiments illustrating our method for multiple types of platforms. We empirically validate the algorithm using a variety of 3D mapping systems, such as a hand-held device, a ground vehicle, and an aerial vehicle. The environments scanned include indoor offices, industrial buildings and open bushland. In all cases, colourisation is done offline but can be done in real-time, such that the processing time is less than the acquisition time.

This paper is organised as follows. Section II provides a short overview of the existing literature on colourising point clouds. Section III outlines an algorithm for temporally synchronising video frames with the device trajectory computed from a 3D mapping device. In Section IV we outline how we determine the points in the point cloud that are visible from a specific camera position. The algorithm used to assign colour to a point from a set of candidates extracted from video is outlined in Section V. The experiments we performed using our proposed algorithm are documented in Section VI. We conclude the paper in Section VII.

II. Related Work

Colourised point clouds are a common output of SLAM algorithms that perceive using a camera (monocular [10], [11], stereo [12]), RGB-D [13] or lidar and camera simultaneously (monocular [14], [15], stereo [16], [17]). The primary focus of these papers, however, is pose and map estimation rather than the coloured point cloud. The colour of each point is typically determined from a single observation i.e. closest frame or first frame. Furthermore, the lidar and camera SLAM algorithms assume that an object measured with lidar can be seen by the camera. This is only universally true for systems where the lidar and camera share the same principal axis. Our approach performs a visibility analysis first and uses only visible observations of a projected 3D point to compute a final colour.

The surveying literature contains many papers where a stationary tripod laser is integrated with a camera in order to acquire detailed geometric and appearance information of a historic or religious site e.g. [18], [19], [20]. The relationship between the camera and the laser scanner can be rigid [19] or non-rigid [18], [20]. A dense surface reconstruction algorithm is used by Moussa et al. [18] to recover fine 3D details not captured by tripod mounted lidar scanners. Point clouds can also be converted to a mesh structure which allows texture information to fill in the space between 3D points in the point cloud [18], [19]. Our algorithm only produces a coloured point cloud where each point is assigned a single colour.

It should be noted that many systems [18], [19], [20] capture high resolution still images as opposed to the video we capture in our work. These systems typically use measured scene information to position each image within the lidar scans. Our method assumes a rigid transformation between the lidar and camera which can be calculated a priori. We also also estimate the temporal offset and scaling which relates timestamps in the device trajectory to timestamps in the captured video.

Identifying points in a point cloud which are visible from a given viewpoint is a very challenging problem as a point cloud contains no surface information in which to identify occlusion. There exists a number of works which attempt to fit a surface to the point cloud assuming that the synthesised surface has certain properties [21]. For example, Xiong et al. [22] recover a triangulated mesh from the point cloud by optimising an $\ell_{2,q}$ objective containing a bi-linear term representing the vertices of a mesh and the edges between vertices. The $\ell_{2,q}$ term was chosen due to its robustness. An alternative approach is to compute point visibility without having to perform a surface reconstruction. The seminal work of Katz et al. [8], [9] defines a hidden point operator which permits the calculation of visible points via a convex hull calculation. The computational complexity of the convex hull calculation is $O(N \log N)$, making it attractive due to the large number of viewpoints in the device trajectory. This algorithm is used in our approach.

III. Temporal Registration

It is important that the camera and mapping device are synchronised before attempting to colourise the point cloud. This is relatively straightforward when both devices share a common clock and triggering. However, this automation is not always practical or available, particularly if the camera has been added to an existing system as is the case in our work. To achieve synchronisation in these circumstances, we assume that the device trajectory $\mathbf{t}$ is computed by the mapping device (which is customary in modern lidar based SLAM algorithms). It is also assumed that $\mathbf{t}$ can be transformed to a camera trajectory using a fixed rigid transformation. The rest of this section details our synchronisation framework.

A. *Camera and lidar Synchronisation*

To synchronise the camera and the lidar data, we correlate the yaw-rate obtained from the camera with the yaw-rate extracted from $\mathbf{t}$. The yaw-rate could also be obtained from an inertial measurement unit (IMU) provided that the raw IMU data is filtered using a complementary filter and the bias is accounted for. Empirical testing has shown that significant yaw motion is present when mapping realistic areas, such that the signal-to-noise ratio (where yaw is the signal) is very high. Roll or pitch (or a linear combination of roll-pitch-yaw) could also be used in scenarios/environments where sufficient yaw motion is not present. For the camera, optical flow information is extracted to compute the yaw-rate. In our implementation, we used the Kanade-Lucas-Tomasi (KLT) feature tracks [23] [24] for the sparse optical flow, although different algorithms can be employed.

The initial estimate of the image timestamps is computed by sampling the device yaw-rate using linear interpolation. We then perform an optimisation to solve for the time shift and rate adjustment that maximises the cross-correlation between the camera yaw-rate and the linearly interpolated device yaw-rate signals. Given that the yaw-rate of a device can be characterised as a high frequency signal with zero-mean and low correlation between consecutive samples (see the red and blue signals in Figures 3b and 3c for an example), cross-correlating the yaw-rates of the camera and $\mathbf{t}$ yields a very high distinct peak, as shown in Figure 3a. This distinctiveness brings high robustness to the temporal synchronisation.

Figures 3b and 3c illustrate an unsuccessful and a successful example of synchronisation between the camera and device yaw-rate. The absolute magnitude of the two yaw-rate signals do not always match, but the relative overlapping of the peaks indicates a good alignment in time. Poor synchronisation can happen due to (*i*) too big of a discrepancy between the start time of the video and lidar data recording or (*ii*) lack of characteristic motion for the lidar/camera resulting in no statistically significant information for optimisation (e.g., an empty featureless corridor). Fortunately, the start time of the video and lidar recording can be controlled and most real environments contain enough features for adequate tracking and synchronisation. Hence the erroneous situation shown in Figure 3b can be avoided in practice.

B. *Spatial Alignment*

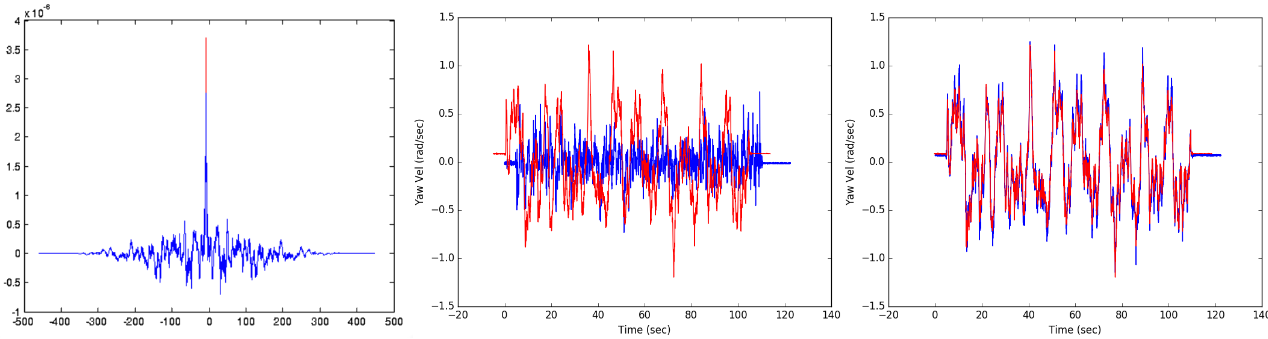
Now that the timestamps of the images are synchronised with the device trajectory, it is now possible to obtain the camera trajectory from the device trajectory. First, the device trajectory is linearly interpolated at the image timestamps to give us the pose of the device at the moment that a given image was captured. The calibrated transform between the device reference frame and the camera (see Section VI-A.1 for details) is then applied to these interpolated poses to yield the pose of the camera when each image was captured.

IV. Selecting Visible Points

As we have already noted, the camera can only observe a subset of the point cloud from a particular position. In this section we outline our approach to identifying the visible points. We start by describing the algorithm proposed by Katz et al. [8], [9].

The camera is assumed to be positioned at the origin of the coordinate system. Each 3D point q_i in the point cloud $Q = [q_1, \ldots, q_N]$ is then mapped to a new 3D point $p_i = F(q_i)$ using the *generalised hidden point removal* operator

$$F(q) = \begin{cases} q\frac{f(\|q\|)}{\|q\|} & q \neq 0 \\ 0 & q = 0 \end{cases} \tag{1}$$



(a) (b) (c)

Fig. 3: Examples of (a) the cross-correlation output, (b) poor time synchronisation, (c) successful time synchronisation. The red and blue signals represent the yaw-rates of the camera and lidar, respectively.

where the function $f : \mathscr{R}^{++} \rightarrow \mathscr{R}^{++}$ is a monotonically decreasing kernel function.

When applying the hidden point removal operator to a point cloud we see that the kernel function f performs a spherical reflection i.e. if $\|q_i\| < \|q_j\|$ then $\|p_i\| > \|p_j\|$. The key insight in the seminal work of [8], [9] is that the visible points in point cloud Q can be found by finding the points that map to points which lie on the convex hull of the transformed point cloud (including the camera as a point in the point cloud as well).

A. Choice of kernel function

The original paper [8] proposed a linear kernel function $f_{\text{linear}}(d;\gamma) = \gamma - d$. With the introduction of the generalised hidden point removal operator [9], the choice of kernel function expanded to include the exponential inversion kernel $f_{\text{exponential}}(d;\gamma) = d^{\gamma}$ with $\gamma < 0$. A key contribution of our paper is to show that these kernels differ with respect to how point visibility changes as a function of an object's scale i.e. how does the choice of kernel impact visibility when a point cloud Q is scaled by $d > 0$? This question is answered in the following two theorems.

Theorem 1: The point visibility algorithm by Katz et al. [8], [9] is scale invariant when using the exponential kernel $f_{\text{exponential}}(d;\gamma)$.

Proof: Let $Q = [q_1, \ldots, q_N]$ be the point cloud in which to compute visibility. Let $V(C)$ be the points that lie on the convex hull of the point cloud C. Let $P(Q;\gamma) = [F(q_1;\gamma), \ldots, F(q_N;\gamma)]$ be the transformation of the point cloud Q using the generalised hidden point removal operator with exponential kernel $f_{\text{exponential}}(d;\gamma)$. The function $V(C)$ is scale invariant, i.e. $V(dC) = dV(C)$, by virtue of the convex hull operator. Since visibility is determined using $V(P(dQ;\gamma))$ we need to show that $V(P(dQ;\gamma)) = g(d)V(P(Q;\gamma))$ or more specifically $P(dQ;\gamma) = g(d)P(Q;\gamma)$ or $F(dq;\gamma) = g(d)F(q;\gamma)$. An expansion of $F(dq;\gamma)$ reveals $F(dq;\gamma) = d^{\gamma}F(q;\gamma)$. ■

Theorem 2: The output of the point visibility algorithm by Katz et al. [8], [9] varies according to the scale of the input point cloud when using the linear kernel $f_{\text{linear}}(d;\gamma)$.

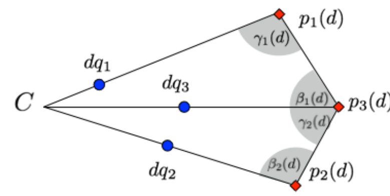


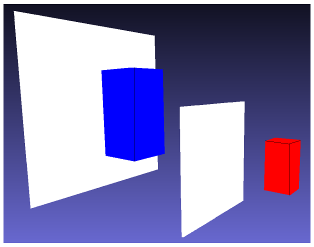
Fig. 4: A point cloud $dQ = d[q_1, q_2, q_3]$ and its spherical reflection $P(dQ;\gamma) = [p_1(d), p_2(d), p_3(d)]$.

.

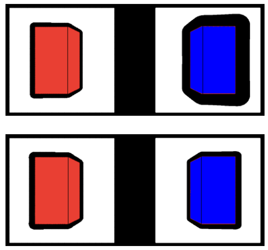
Proof: The proof of this theorem is quite involved and is postponed to Appendix B. ■

The fact that the computation of visibility using the linear kernel is affected by an object's scale is clearly an undesirable property. We now illustrate where this property manifests itself in practice. In Figure 4 we see a point cloud of a concave structure $dQ = d[q_1, q_2, q_3]$ where the point dq_1 belongs to a foreground object and the points dq_2 and dq_3 represent a background object. We assume that for $d = 1$, the points Q are classified as visible. The proof for Theorem 2 shows that the angles $\beta_1(d)$ and $\gamma_2(d)$ formed in the transformed point cloud $P(dQ;\gamma) = [p_1(d), p_2(d), p_3(d)]$ are monotonically increasing functions of d when $\|q_1\| < \|q_2\| < \|q_3\|$. Therefore, there exists a d such that $\beta_1(d) + \gamma_2(d) > \pi$, causing the point $p_3(d)$ to no longer lie on the convex hull and thus the point dq_3 is classified as invisible.

A synthetic example illustrating this property is shown in Figure 5. An extremely dense point cloud is created by first placing a red 3D box in front of a white planar wall. This structure is then duplicated (blue coloured box), scaled by $d > 1$ and then translated such that when the combined structures (Figure 5a) are imaged with perfect visibility analysis, a symmetric image is produced. Figure 5b shows the resulting projections of the combined structure using the linear (top) and exponential inversion (bottom) kernels for determining visibility. The black pixels in the projections correspond to empty space. As predicted, the increase in scale has adversely impacted the visibility analysis when using the linear kernel. The change in scale has had no effect



(a)



(b)

Fig. 5: A synthetic scene (a) is used to assess scale invariant properties of two kernels proposed in [9]. Figure (b) shows the projection of the visible points in (a) computed using the linear (top) and exponential inversion (bottom) kernels.

when using the exponential kernel.

The scale invariant property and the associated analysis of concave structures makes the exponential kernel attractive for determining point visibility within point clouds containing vast numbers of objects as is the case when using mobile mapping devices.

B. Computational Considerations

The device trajectory computed by the camera-less 3D mapping device contains the pose of the device as a function of time. Every change in pose requires computing the visible points using an $O(N \log N)$ operation [8]. Point clouds created using lidar based mapping devices contain many millions of points, making it prohibitively slow to perform an $O(N \log N)$ operation for every pose in the device trajectory. To alleviate this problem we employ two relaxations.

The first relaxation involves selecting points which are within a hemisphere of radius R_m of the camera position. This operation can be performed efficiently by discretising the point cloud into voxels with side length V and computing an exemplar for each voxel. Points are selected from the point cloud if their associated exemplar lies inside the hemisphere.

The second relaxation requires introducing the assumption that nearby points have the same visibility. This is reasonable provided that the distance to nearby points is not too large. Visibility can therefore be computed by discretising the point cloud as before, determining the visibility of each voxel exemplar and then propagating the result to the points represented by the exemplar.

An important property of both relaxations is that the computational complexity of each step is now upper bounded by the number of voxels. In our experiments we compute the exemplar point by finding the corner of the voxel which is closest to the 3D point at negative infinity (i.e. $-[\infty, \infty, \infty]$) which can be computed efficiently using the floor operator.

V. Point Colouring

It is now possible to extract a set of candidate colours for each 3D point in the point cloud using the extrinsic calibration between the camera and the device, the device trajectory $\mathbf{t}$ and the visibility analysis. This section describes how we reduce the set of colour candidates to a single colour.

The number of points in a point cloud is typically very large, making it intractable to store every candidate colour for every 3D point in memory. To render the problem tractable, we use a sequential method that is robust to outliers. We assume that the point cloud quality from SLAM is of the same order as current market products such as GeoSLAM, Kaarta and Google Cartographer, which typically have few centimeters of positional noise. Colourisation errors due to small inaccuracies in map, camera pose and timing are reduced by using a form of robust average that can be calculated sequentially. We do this by estimating the mean and covariance of a weighted Gaussian distribution over the set of colours. The final colour assigned to the 3D point is the mean (μ) of this estimated distribution.

Consider the problem of estimating the mean and covariance of a weighted Gaussian distribution using the following log likelihood function

$$\arg\max_{\mu,\Sigma} \sum_{i=1}^{N} w_i \log \mathcal{N}(x_i; \mu, \Sigma) \quad (2)$$

where $\mathcal{N}$ is the multivariate Gaussian density function, x_i is a candidate colour and $w_i \geq 0$ is a weight assigned to the colour x_i. The solution for μ and Σ is

$$\mu = \frac{w_N x_N + \sum_{i=1}^{N-1} w_i x_i}{w_N + \sum_{i=1}^{N-1} w_i} \qquad \Sigma = \frac{w_N S_N + \sum_{i=1}^{N-1} w_i S_i}{w_N + \sum_{i=1}^{N-1} w_i} \quad (3)$$

where $S_i = (x_i - \mu)(x_i - \mu)^T$. We see from (3) that the contributions of the previous $N-1$ colour candidates can be represented using three quantities: $\hat{w} = \sum_{i=1}^{N-1} w_i$, $\hat{\mu} = \sum_{i=1}^{N-1} w_i x_i$ and $\hat{\Sigma} = \sum_{i=1}^{N-1} w_i S_i$. Thus, each point in the point cloud requires three state variables during processing. Critically, the video recorded by the camera can now be processed sequentially. For this paper, the weights w_i are computed using an unweighted Gaussian distribution

$$\mathcal{N}\left(x; \frac{\sum_{i=1}^{N-1} x_i}{N-1}, \frac{\sum_{i=1}^{N-1} S_i}{N-1}\right) \quad (4)$$

This choice of weighting function provides a balance between robustness to outlier colours and consistency with respect to the order in which observations arrive. This function requires an additional two state variables per point. The memory required for the state variables is much less than the memory required to store all colour candidates.

In addition to this rolling robust average method, we also proportionally weight each colour candidate according to the 3D point's distance to the camera. This weighting reflects the reduction in certainty of the pixel location with increasing camera distance when there is angular uncertainty. It also exploits the fact that SLAM systems normally have higher local accuracy spatially, and most inaccuracy is at the large scale due to accumulated drift. Preferencing the closer observations aids in both cases.

VI. Experiments

In all experiments we used existing camera-less lidar scanning devices and added our cameras to them. This section provides implementation details and results.

A. Practical Considerations

The system runs on a Mid 2014 MacBook Pro with Intel Core i7 CPU @ 2.20GHz with four cores and 16GB of memory. We used two types of consumer cameras in our tests: a GoPro 4 Session and a Ricoh Theta S 360° camera. Both cameras record video at 29.97 frames per second. Three types of platforms were used for testing: a hand-held device (Figure 1b) built in-house by our team, an unmanned aerial vehicle DJI Matrice 600 (Figure 1d), and an electric all-terrain John Deere TE Gator autonomous ground vehicle (Figure 1e). One of the goals of the multiple platforms is to illustrate the easy applicability and portability of the system, as shown in the very different setups in the pictures. To generate $\mathbf{t}$ and the 3D point cloud to be colourised, we use the SLAM implementation described in [1] and [2].

1) Extrinsic Calibration: The objective of the calibration step is to determine the extrinsic transformation between the lidar's base reference frame and the camera. To this end, we have implemented a visual tool that allows the user to manually adjust the view of the point cloud over the camera image. To calibrate the extrinsics, the user tunes every component of the transformation (translation, rotation and scale) until the required degree of accuracy is obtained. The quality of the calibration is evaluated by performing a perspective projection of 3D points visible by the camera to the image plane and observing the quality of the alignment between features that are distinctive enough in both modalities.

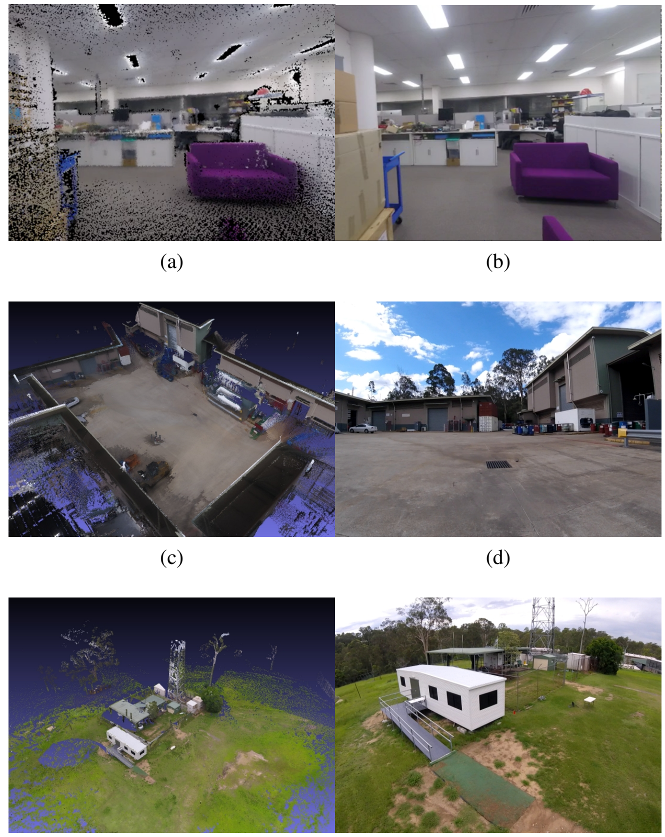
2) Key Parameters: As discussed throughout the paper, several parameters affect the quality and processing time of the system. In our experiments, we present results with different parameter configurations, summarised in Table I.

B. Results and Discussion

1) Hand-held: The hand-held device is equipped with a Hokuyo UTM-30LX, which has a 30 meter range. We ran tests in multiple areas such as indoor offices, corridors and industrial environments, recording the data at walking speed. A snapshot of the colourised point cloud of our office environment and the corresponding camera view is shown in Figures 6a and 6b, respectively. In this type of environment, the visibility check (Section IV) brought significant visual improvements due to the more cluttered nature of the space.

2) Ground Vehicle: The ground vehicle was driven at approximately 2m/s, in an industrial park (Figure 6c). As illustrated in Figure 1e, there is a significant translation from the lidar to the camera, necessitating the use of visibility analysis. The resulting point cloud is adequately colourised despite significant vibration of the lidar mounting post. This platform used the Velodyne VLP-16 lidar, which has a 100m range. In this case, we used only 4 of the 16 beams available, which lead to faster than real-time processing.

3) Aerial Vehicle: The aerial platform also employs the Velodyne VLP-16 lidar. The camera mounting is once again different, and given the size and limited payload capacity of the quad-copter, the addition of a small camera without the need for extra cabling or processing is convenient. Figures 6e and 6f show the results.



(a) (b) (c) (d) (e) (f)

Fig. 6: Resultant colourised point clouds for (a,b) a hand-held system, (c,d) a ground vehicle, and (e,f) an aerial vehicle. The left column shows a view of the coloured point clouds and the right column shows captured camera frames.

Fig. 7: Example illustrating the misregistration in areas of sharp 3D discontinuity and colour variation, such as the brown roof line against the blue sky (detail shown in the right image, corresponding to the green box on the left).

In addition to the results presented in this section, point clouds are available online[1]. We have also created a summary video[2].

4) Discussion: The quality of the coloured point clouds produced by our method is adversely impacted by errors in the device trajectory computed by the SLAM algorithm,

[1] https://doi.org/10.4225/08/5afb6a9e8a9f1 - The point clouds are in .ply format and can be visualised using free software (e.g., Meshlab).

[2] https://youtu.be/7LeQUtlYtHU

TABLE I: System parameters. The ‘Values’ column shows typical values, depending on the point density and speed required.

Parameter Name	Description	Values	Comments
Point Skip (P_s)	Amount of decimation in the original point cloud	1, 5, 9 ...	Affects the processing time with n^2
Frame Skip (F_s)	Extract colour candidates using every F_s-th frame	1, 5, 9 ...	Affects the processing time linearly
Maximum Range (R_m)	The radius of the hemisphere used for visibility determination	>6m	See Section IV.
Voxel side length (V)	Voxel side length for visibility determination	0.05m	See Section IV
Kernel Type	Choice of kernel function to perform radial inversion during the visiblity check	Linear Exponential	Scale invariability (see Section IV)
γ	Visibility kernel parameter that determines the size of the region detected as visible	$\gamma_{linear} < \max_{p_i \in P}(\|p_i\|)$ $\gamma_{exp} < 0$	See Section IV

TABLE II: Camera yaw-rate and colour estimation processing times for the datasets shown in Figure 6. For all cases, the ‘Kernel Type’ was the exponential inversion kernel with $\gamma = -0.001$.

Dataset	Acquisition	Input Points	[F_s, R_m]	Yaw-rate	Colourisation
Hand-held	4:26	4,212,425	[30, 7]	2:37	3:18
Ground	2:03	21,351,584	[30, 35]	0:39	11:59
Aerial	5:26	3,536,796	[30, 60]	4:31	11:51

TABLE III: Average root mean square error (RMSE) between the estimated point colours and associated candidate colours for different datasets and algorithm configurations.

Device	Exp Kernel (γ)	Avg RMSE	Pts Coloured %
Hand-held	-0.0001	38.58	82.05
Hand-held	-0.001	33.75	63.24
Hand-held	-0.01	27.37	29.39
Ground	-0.0001	34.90	75.35
Ground	-0.001	30.83	34.53
Ground	-0.01	24.50	11.19
Aerial	-0.0001	33.89	96.06
Aerial	-0.001	35.44	59.67
Aerial	-0.01	36.82	15.06

camera distortion parameters, camera intrinsics, device to camera extrinsics, camera and lidar synchronisation, visible point classification, and the robustness of the algorithm used to select a colour from a set of candidate colours. In this section we discuss the last three sources of error. Our experiments did show a misalignment between the video frame and projected point cloud data in areas with high frequency motion in conjunction with a sharp depth discontinuity. It is difficult to determine if the misalignments were due to synchronisation issues or the sub-optimal procedure used for estimating extrinsics. A visual inspection of the cross correlation of the camera yaw-rate signals with their respective device yaw-rate signals showed very high peaks, indicating successful synchronisation. The effect of the misalignment can be seen in Figure 7 where a brown roof line has been coloured with pixels belonging to blue sky. Another source of errors are glass and reflective surfaces. This is inherent to the lidar itself (and not directly due to the colourisation algorithm), but it does affect the colourisation quality.

Table III shows how the gamma parameter of the exponential inversion kernel affects the number of points coloured and the robustness of the point colouring algorithm. The percentage of points which were classified as being visible in one or more video frames is shown in the “Pts Coloured” column. The average root mean square error between the point colour and the candidate colours is found in the “Avg RMSE” column. As predicted, more points are classified as visible as the γ parameter approaches zero. Results also show that the average RMSE error increases as the percentage of points coloured increases in all datasets except the aerial vehicle dataset.

The processing time required to process each dataset is shown in Table II. The duration of the captured video, the number of points to be coloured and the parameters used to colour each dataset is included in the table. The “Yaw-rate” column contains the amount of time required to compute the yaw-rate from the captured video. The “Colourisation” column contains the amount of time required to perform visibility analysis over the entire sequence and compute the final colour of each point. We see that the bulk of the processing time is spent in the colourisation phase.

VII. Conclusion

We have presented an approach to colourising a point cloud acquired with a mobile scanning platform that does not require tight coupling of a mapping device and camera either spatially or temporally. To this end, we introduced a novel method of synchronising the mapping device data and camera data using optical flow information. The newly-proposed colourisation pipeline integrates the state-of-the-art point cloud visibility analysis algorithm, for which we have motivated the specific choice of kernel theoretically and empirically. The colour accumulation and assignment scheme employed by our pipeline is both memory-efficient and robust to outliers resulting from variations in lighting conditions or local misalignment between the mapping device and camera. Finally, we have demonstrated the flexibility of our colourisation pipeline by applying it to data recorded using variety of different scanning platforms, be it hand-held, autonomous ground vehicle, or aerial vehicle. Future work includes adding closed-loop adjustment of the alignment between the lidar and camera data. This would mitigate the problem of colours ‘bleeding’ onto adjacent 3D structures and would further improve the quality of colourisation.

APPENDIX

A. Linear Kernel and Monotonic Angle Functions

Lemma 1: The angle $\gamma_1(d)$ in Figure 4 is monotonically decreasing if the transformed points $[p_1(d), p_3(d)]$ are calculated using the linear kernel $p_i(d) = (\lambda\|q_i\|^{-1} - d)q_i$ with $\|q_3\| > \|q_1\|$ and $\lambda\|q_i\|^{-1} - d > 0$.

Proof: We remark that the angle $\alpha_1 = \pi - \beta_1(d) - \gamma_1(d)$ is constant for all d. Using the law of sines we have

$$f(d) = \frac{\|p_1(d)\|}{\|p_3(d)\|} = \frac{\sin(\pi - \alpha_1 - \gamma_1(d))}{\sin(\gamma_1(d))}. \quad (5)$$

The derivative of (5) is $f'(d) = -\sin(\alpha_1)\csc^2(\gamma_1(d))\gamma_1'(d)$ $\gamma_1(d)$ is monotonically decreasing if $f'(d) > 0$ for all d. Substituting $p_i(d)$ in to Equation 5 gives

$$f(d) = \frac{\lambda - d\|q_1\|}{\lambda - d\|q_3\|}. \quad (6)$$

An analysis of the inequality $f'(d) > 0$ where $f'(d)$ is the derivative of Equation 6 results in $\|q_3\| > \|q_1\|$ which is true by definition. ■

As consequence of this result, $\beta_1(d)$ is monotonically increasing since $\pi = \alpha_1 + \gamma_1(d) + \beta_1(d)$. This approach proves $\gamma_1(d)$ increases monotonically when $\|q_3\| < \|q_1\|$.

B. Linear Kernel Scale Dependent Visibility Proof

This section documents the proof for Theorem 2 i.e. the algorithm proposed by Katz et al. [8], [9] for computing visibility of points in a point cloud about a position C is not scale invariant when using the linear kernel. *Proof:* Figure 4 shows a point cloud $\{dq_1, dq_2, dq_3\}$ with $\|q_1\| < \|q_2\| < \|q_3\|$ where all points are classified as visible when $d = 1$. By definition, the reflected points $p_i(d) = (\lambda\|q_i\|^{-1} - d)q_i$ therefore lie on the convex-hull when $d = 1$ and the angles $\beta_1(1)$ and $\gamma_2(1)$ satisfy $\beta_1(1) + \gamma_2(1) \leq \pi$. According to Lemma A, the angles $\beta_1(d)$ and $\gamma_2(d)$ are monotonically increasing functions of d since $\|q_1\| < \|q_3\|$ and $\|q_2\| < \|q_3\|$ respectively. It is therefore possible to select a scaling of the structure d and parameter λ such that $\beta_1(d) + \gamma_2(d) > \pi$ and $\lambda\|q_i\|^{-1} > d$, rendering the point dq_3 invisible. ■

ACKNOWLEDGEMENT

The authors would like to thank the Hovermap Team [6] for the experiments with the aerial platform.

REFERENCES

[1] M. Bosse and R. Zlot, "Continuous 3d scan-matching with a spinning 2d laser," in *IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 4312–4319.

[2] M. Bosse, R. Zlot, and P. Flick, "Zebedee: Design of a spring-mounted 3-d range sensor with application to mobile mapping," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1104–1119, Oct 2012.

[3] R. Dubé, A. Gawel, H. Sommer, J. Nieto, R. Siegwart, and C. Cadena, "An online multi-robot slam system for 3d lidars," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[4] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time." in *Robotics: Science and Systems*, vol. 2, 2014.

[5] GeoSLAM. (2018) Zebcam. [Online]. Available: https://geoslam.com/wp-content/uploads/2017/08/GeoSLAM-ZEB-CAM.pdf

[6] Emesent. (2018) Hovermap payload. [Online]. Available: http://http://www.emesent.io

[7] A. Pfrunder, P. V. K. Borges, A. R. Romero, G. Catt, and A. Elfes, "Real-time autonomous ground vehicle navigation in heterogeneous environments using a 3d lidar," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept 2017, pp. 2601–2608.

[8] S. Katz, A. Tal, and R. Basri, "Direct visibility of point sets," in *ACM SIGGRAPH 2007 Papers*, ser. SIGGRAPH '07. New York, NY, USA: ACM, 2007. [Online]. Available: http://doi.acm.org/10.1145/1275808.1276407

[9] S. Katz and A. Tal, "On the visibility of point clouds," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec 2015, pp. 1350–1358.

[10] 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.

[11] W. N. Greene, K. Ok, P. Lommel, and N. Roy, "Multi-level mapping: Real-time dense monocular slam," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 833–840.

[12] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.

[13] B. W. Babu, S. Kim, Z. Yan, and L. Ren, "σ-dvo: Sensor noise model meets dense visual odometry," in *Mixed and Augmented Reality (ISMAR), 2016 IEEE International Symposium on*. IEEE, 2016, pp. 18–26.

[14] J. Zhang and S. Singh, "Visual-lidar odometry and mapping: Low-drift, robust, and fast," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 2174–2181.

[15] T. Lowe, S. Kim, and M. Cox, "Complementary perception for handheld slam," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1104–1111, 2018.

[16] W. Maddern and P. Newman, "Real-time probabilistic fusion of sparse 3d lidar and dense stereo," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016, pp. 2181–2188.

[17] H. Alismail, L. D. Baker, and B. Browning, "Continuous trajectory estimation for 3d slam from actuated lidar," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 6096–6101.

[18] W. Moussa, M. Abdel-Wahab, and D. Fritsch, "Automatic fusion of digital images and laser scanner data for heritage preservation," in *Progress in Cultural Heritage Preservation*, M. Ioannides, D. Fritsch, J. Leissner, R. Davies, F. Remondino, and R. Caffo, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 76–85.

[19] W. Neubauer, M. Doneus, N. Studnicka, and J. Riegl, "Combined high resolution laser scanning and photogrammetrical documentation of the pyramids at giza," in *CIPA XX International Symposium*, 2005, pp. 470–475.

[20] A. Abdelhafiz, B. Riedel, and W. Niemeier, "Towards a 3d true colored space by the fusion of laser scanner point cloud and digital photos," in *Proceedings of the ISPRS Working Group V/4 Workshop (3D-ARCH*, 2005.

[21] M. Berger, A. Tagliasacchi, L. Seversky, P. Alliez, J. Levine, A. Sharf, and C. Silva, "State of the art in surface reconstruction from point clouds," in *EUROGRAPHICS star reports*, vol. 1, no. 1, 2014, pp. 161–185.

[22] S. Xiong, J. Zhang, J. Zheng, J. Cai, and L. Liu, "Robust surface reconstruction via dictionary learning," *ACM Trans. Graph.*, vol. 33, no. 6, pp. 201:1–201:12, Nov. 2014. [Online]. Available: http://doi.acm.org/10.1145/2661229.2661263

[23] C. Tomasi and T. Kanade, "Detection and tracking of point features," in *International Journal of Computer Vision*, vol. 9, 01 1991, pp. 137–154.

[24] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision (darpa)," in *Proceedings of the 1981 DARPA Image Understanding Workshop*, April 1981, pp. 121–130.