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.

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 $t$, 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 $t$ 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
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 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 $t$ is computed by the mapping device (which is customary in modern lidar based SLAM algorithms). It is also assumed that $t$ 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 $t$. 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 $t$ 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

$$F(q) = \left\{ \begin{array}{ll} \frac{f(||q||)}{||q||} q \\ 0 & q = 0 \end{array} \right. (1)$$
where the function \( f : \mathbb{R}^+ \to \mathbb{R}^+ \) 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. 3](image1.png)

**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.

![Fig. 4](image2.png)

**Fig. 4:** A point cloud \( dQ = [q_1, q_2, q_3] \) 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 \( dQ = [q_1, q_2, q_3] \) 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...
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 (\(\mu\)) of this estimated distribution.

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

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

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 \(\mu\) and \(\Sigma\) is

\[
\mu = \frac{w_N x_N + \sum_{i=1}^{N-1} w_i x_i}{w_N + \sum_{i=1}^{N-1} w_i}, \quad \Sigma = \frac{w_N S_N + \sum_{i=1}^{N-1} w_i S_i}{w_N + \sum_{i=1}^{N-1} w_i}
\]  

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 \(\hat{S} = \sum_{i=1}^{N-1} w_i S_i\). 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

\[
\mathcal{N}\left(x, \mu_0, \Sigma_0\right), \quad w_i = \frac{\sum_{j=1}^{N-1} x_j \cdot x}{N-1}, \quad \frac{\sum_{j=1}^{N-1} S_j}{N-1}
\]  

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 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.

In addition to the results presented in this section, point clouds are available online1. We have also created a summary video2.

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.

---

1https://doi.org/10.4225/08/5af6a0e8a0f1 - The point clouds are in .ply format and can be visualised using free software (e.g., Meshlab).
2https://youtu.be/7LyQ0rlHt5U
TABLE I: System parameters. The ‘Values’ column shows typical values, depending on the point density and speed required.

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

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 $\gamma = -0.001$.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Acquisition</th>
<th>Input Points</th>
<th>[F, Rm]</th>
<th>Yaw-rate</th>
<th>Colourisation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ground</td>
<td>2:03</td>
<td>21,351,584</td>
<td>[30, 35]</td>
<td>0:39</td>
<td>11:59</td>
</tr>
<tr>
<td>Aerial</td>
<td>5:26</td>
<td>3,536,796</td>
<td>[30, 60]</td>
<td>4:31</td>
<td>11:51</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Device</th>
<th>Exp Kernel ($\gamma$)</th>
<th>Avg RMSE</th>
<th>Pts Coloured %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hand-held</td>
<td>-0.0001</td>
<td>38.58</td>
<td>82.05</td>
</tr>
<tr>
<td>Hand-held</td>
<td>-0.001</td>
<td>33.75</td>
<td>63.24</td>
</tr>
<tr>
<td>Hand-held</td>
<td>-0.01</td>
<td>27.37</td>
<td>29.39</td>
</tr>
<tr>
<td>Ground</td>
<td>-0.0001</td>
<td>34.90</td>
<td>75.35</td>
</tr>
<tr>
<td>Ground</td>
<td>-0.001</td>
<td>30.83</td>
<td>34.53</td>
</tr>
<tr>
<td>Ground</td>
<td>-0.01</td>
<td>24.50</td>
<td>11.19</td>
</tr>
<tr>
<td>Aerial</td>
<td>-0.0001</td>
<td>33.89</td>
<td>96.06</td>
</tr>
<tr>
<td>Aerial</td>
<td>-0.001</td>
<td>35.44</td>
<td>59.67</td>
</tr>
<tr>
<td>Aerial</td>
<td>-0.01</td>
<td>36.82</td>
<td>15.06</td>
</tr>
</tbody>
</table>

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 $\gamma_i(d)$ in Figure 4 is monotonically decreasing if the transformed points $\{p_1(d), p_3(d)\}$ are calculated using the linear kernel $p_i(d) = (\lambda_i \|q_i\|-d)q_i$ with $\|q_3\| > \|q_1\|$ and $\lambda_i \|q_i\|-d > 0$.

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

$$f(d) = \frac{\|p_1(d)\|}{\|p_3(d)\|} = \frac{\sin(\pi - \alpha_i - \gamma_i(d))}{\sin(\gamma_i(d))}.$$  

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

$$f(d) = \frac{\lambda_i - d\|q_1\|}{\lambda_i - d\|q_3\|}.$$  

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, $\beta_i(d)$ is monotonically increasing since $\pi = \alpha_i + \gamma_i(d) + \beta_i(d)$. This approach proves $\gamma_i(d)$ 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 $p_i(d) = (\lambda_i \|q_i\|-d)q_i$ therefore lie on the convex-hull when $d = 1$ and the angles $\beta_i(1)$ and $\gamma_i(1)$ satisfy $\beta_i(1) + \gamma_i(1) \leq \pi$. According to Lemma A, the angles $\beta_i(d)$ and $\gamma_i(d)$ 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 $\lambda$ such that $\beta_i(d) + \gamma_i(d) > \pi$ and $\lambda\|q_i\|-d > 0$, 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