PaintPath: Defining Path Directionality in Maps for Autonomous Ground Vehicles

Riley Bowyer[1], Thomas Lowe[1], Paulo Borges[1], Tirthankar Bandyopadhyay[1], Tobias Löw[1,2], David Haddon[1]

***Abstract*— Directionality in path planning is essential for efficient autonomous navigation in a number of real-world environments. In many map-based navigation scenarios, the viable path from a given point A to point B is not the same as the viable path from B to A. We present a method that automatically incorporates preferred navigation directionality into a path planning costmap. This 'preference' is represented by coloured paths in the costmap. The colourisation is obtained based on an analysis of the driving trajectory generated by the robot as it navigates through the environment. Hence, our method augments this driving trajectory by intelligently colouring it according to the orientation of the robot during the run. Creating an analogy between the vehicle orientation angle and the hue angle in the Hue-Saturation-Value colour space, the method uses the hue, saturation and value components to encode the direction, directionality and scalar cost, respectively, into a costmap image. We describe a costing function to be used by the A* algorithm to incorporate this information to plan direction-aware vehicle paths. Our experiments with LiDAR-based localisation and autonomous driving in real environments illustrate the applicability of the method.**

I. INTRODUCTION

Path planning is an essential element of autonomous vehicles. It defines, according to arbitrary criteria, the optimal (or near-optimal) path that a vehicle should follow to go from a starting point to an end-goal. In practice, the task is usually achieved by considering two main elements, namely *global* path planning and *local* path planning. The global plan provides the full and overall path based on a known map $\mathcal{M}$ of the environment, with the assumption that the world is static. As the vehicle moves through the global path, it often encounters moving obstacles or changes in the world (as identified by its on-board sensors) that were not originally incorporated in $\mathcal{M}$. The local planner ideally avoids such obstacles and adapts the vehicle's path causing it to deviate from its original plan, however aiming to return to the global path as soon as the environmental circumstances allow.

Most traditional path planning algorithms are suited to scenarios in which the cost of travel is independent of the direction of travel. In other words, the path planned to go from a point A to a point B is identical to the path from B to A. In most practical applications, however, discriminating the direction of travel is key to successful operations. On urban networks, for example, there exist a number of rules regarding lane directionality in both one-way and two-way roads. Not respecting those rules can obviously lead to accidents. Challenges also exist in 3D off-road terrains, where a vehicle is able, for instance, to go downhill but not uphill on a given slope. In industrial areas or plants, there are frequently traffic rules for machinery (e.g., forklifts, trucks) and pedestrian areas. A strategy to efficiently incorporate many of those directionality constraints in a map is the core contribution of this paper.

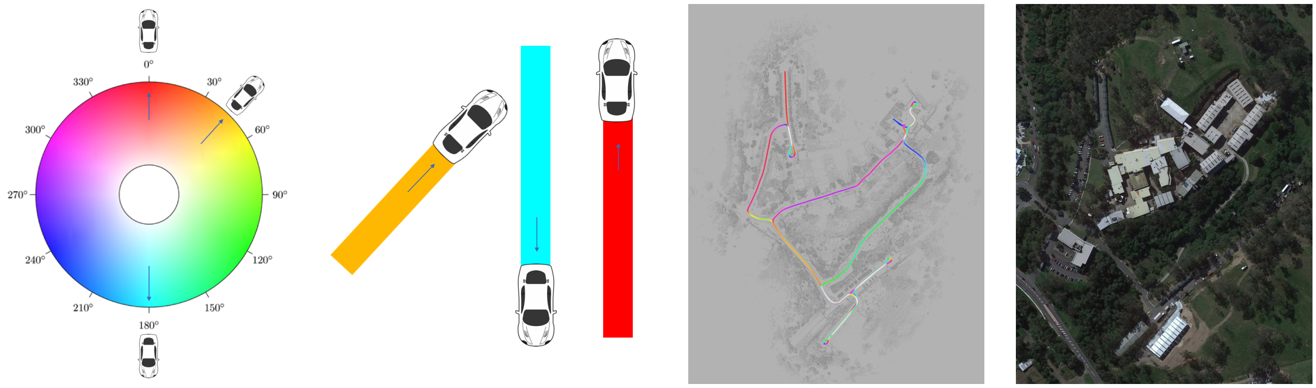
As mentioned, global planners operate on a known map $\mathcal{M}$. When navigating through $\mathcal{M}$ the vehicle navigates only through areas that are feasible, either because of physical or legislation constraints. Based on this inherent characteristic of how/where vehicles must travel, we present a novel method to extract information from the vehicle trajectory (assuming the vehicle has a localisation system that provides such trajectory) to incorporate preferred directionality into the maps. A key challenge lies on how to effectively represent the directionality in a way that can be both mathematically interpreted by the path planning algorithm but also intuitively understood by human observers of that map.

As two-dimensional path planning algorithms for ground vehicles often analyse the costmap represented in an image data structure, the gist of the proposed method is to represent directionality as different colours in a map. The method, coined PaintPath, colours traversed paths according to the two-dimensional orientation of the sensor during navigation, where the orientation can be in the range $[0, 2\pi]$. This colour encoding is done in the HSV colour space, in which the *hue* (H) component also has a circular representation falling in the range $[0, 2\pi]$. The concept is illustrated in Figure 1, where a costmap is coloured according to traversed directions.

Hence, the hue can be used as rule for in which direction a vehicle should go when traversing through a path. In addition, in order to bring representation completeness to PaintPath, we also use the *saturation* (S) and *value* (V) components. The saturation defines how important directionality is in a particular path (e.g., there are areas in which a vehicle can go in both directions without consequences and other areas where it is absolutely forbidden to go in a given way). A low directionality importance is illustrated in the 'whitish' path segments in Figure 1c. In those segments, the vehicle traversed in multiple directions, therefore indicating the 'two-way' nature of that path, consequently lowering the saturation. Finally, the value component represents the overall cost in the map, incorporating obstacles (i.e., a low value shown in black represents an obstacle).

For planning on the coloured representation, we have mod-

[1]Robotics and Autonomous Systems Group, Data61, CSIRO, Brisbane, Australia
[2]Multi-Scale Robotics Lab, ETH Zürich, Zurich, Switzerland,(tobias.loew@alumni.ethz.ch)
{First.LastName}@csiro.au



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

Fig. 1: Illustration of the PaintPath hue mapping. Figure (a) shows the HSV hue associated to vehicles travelling in different directions. Those direction hues are painted onto the vehicle paths, as conceptually depicted in (b). The paths are transferred to a real map (c), where all the coloured paths were traversed by the vehicle. Figure (d) is the satellite image of the costmap given in (c), shown only for reference. Please note that this figure is only illustrating the 'hue' component of the algorithm. The role of the other elements (saturation, value) are discussed in detail in Section III.

ified an A* algorithm such that the mathematical formulation incorporates the 'HSV' costs in the optimisation, rather than only the free vs non-free space cost. The proposed method adds minimal computational effort to the map generation process. Regarding planning, it increases the computational complexity depending on the number of colour channels considered (the planner analyses three colour channels instead of one).

We present experiments in multiple scenarios illustrating the applicability of the method for ground vehicle navigation. Although PaintPath is not constrained to any specific localisation system or sensing modality, in our experiments we use a LiDAR-inertial algorithm based on the continuous-time SLAM implementation [1], as it is suitable for autonomous vehicle operation environments [2].

This paper is organised as follows. In Section II we review relevant literature, contrasting with the proposed method. In Section III we detail PaintPath. In Section IV we present a number of experiments illustrating the applicability of the method followed by relevant conclusions in Section V.

II. Literature Review

A common problem decomposition for large scale unmanned ground vehicles (UGVs) is the breakdown of a high level global planner with simplified vehicle characteristics that prescribes a path to follow and a local planner that executes the path as best as it can, incorporating more realistic vehicle model, terrain traction constraints, and non-holonomic and steering constraints. The local planner also responds to local obstacles and other real time sensory feedback. Here we focus on the large scale global planning rather than local planners.

Due to the nature of traffic, directionality in path planning is critical for self driving cars. The directionality constraints in such scenarios are encoded as a directed graph representation with edge weights and an optimal path is found by running a minimum cost path search on such road networks [3]. At a street level scale, road lane graphs are utilised to guide the self driving vehicle on the directionality of motion. These lane graphs are partially algorithmically generated from higher-level street network map and partly human edited [4]. These approaches soon become intractable in semi-structured and un-structured environments.

For unstructured rough terrains, the most popular approach is to generate a costed grid map, run A* [5] or Dijkstra's algorithm [6] on the maps for the global path and run more sophisticated model based local planners for online adaptation.

Often due to the simplification of real constraints and the abstraction of the problem, the prescribed global path generates paths that the local planner is unable to execute. An example is the terrain influenced by directionality constraints. As an example, the vehicle control and traction might only allow the vehicle to go in one direction, however getting stuck or losing traction in other directions from the same position.

Global navigation functions [7] are able to encode directionality into the map. Often this requires formulating a cost or a value function over the map that is then used to generate a navigation policy at each position. However, coming up with such cost functions is challenging as the contributors to the costs are always not measurable (effect of traction in mobility) and have coupled non-linear effects.

Beyond grid based formulation, some approaches try to solve the full 3-D problem by combination of RRT* [8] and local trajectory optimisation that incorporates 3D terrain shape and kinematic vehicle constraints purely on 3D pointclouds [9], or a variant in 2.5D simplification as DEM (elevation maps) [10], [11]. Approaches with simplification of meshes have also been proposed [12], [13]. Direction dependent tip over constraints has been considered in [14]. In most of these approaches, directionality is not explicitly captured in the maps requiring an evaluation of a suitable local trajectory at run-time, potentially leading to unsuitable solutions.

Integrating non-holonomic constraints explicitly into the poly-linear global path has been done by adapting to smoothen a low dimensional path [15]. State Lattice planner for global path planning using rough terrain trajectory generation for connecting states in lattice has also been proposed [16], allowing for the incorporation of kinodynamic constraints into the global paths.

Different representations of the map like isolines-based maps [17] and modelling terrains as cubic B-Patches [18] can be useful in a number of scenarios, however they only capture the structure of the environment rather than the traversability or directionality. It is also possible to observe the flow of dynamic objects (e.g., people and other vehicles) in the environment [19] to incorporate 'directional flow' in the planning. However, this is a probabilistic process that requires long term observation of the area.

Traversability approaches are either appearance based using combination of visual and laser sensor [20] or a scalar binary or continuous value that is learned offline [21] or online [22]. An A* or trajectory optimization planners like Time Elastic Bands [23], RRT* or STOMP [24] is subsequently run on such maps to produce optimized paths. These approaches are restrictive as traversability values do not often capture the local dynamics in the surface patches. As an example, traversability in one direction may be completely different from another direction, especially in the case of sloped surfaces. The component of directional dependent traversability has not been explored significantly in the literature.

To address the limitation of robust driving on challenging terrains, analysis of expert human driven trajectories have been explored. There has been work in learning from human drivers and human GPS input for autonomous vehicles control and path planning. Learning steering function for vehicle from human driver is explored by using neural networks [25] and reinforcement learning [26]. Closest to our work is that by Ziebart et al [27] and Choi and Kim [28], based inverse reinforcement learning for learning taxi drivers preference for road segments from GPS trace data. However there has been limited work in modeling direction driven route preferences in rough terrains and outside of road networks (e.g., plants, private complexes, etc) from human drivers. This is the core element that the PaintPath method addresses.

III. Proposed Method

In this section we explain in detail the PaintPath algorithm. We also describe the concept of navigation map restriction, which serves as a basis for PaintPath.

A. Map Restrictor

When navigating in pre-mapped areas to perform a given task it is reasonable to assume that in many cases the most desired path to be taken by the vehicle corresponds to a previously traversed path. With this assumption, a path of low cost can be added to a traditional costmap to encourage a planner to follow previously driven routes. This strategy relies upon the generation of a trajectory list in addition to any other requirements for the generation of a costmap. Biasing a planner to a previously driven path increases the likelihood that an efficient and safe route is taken without fully restricting the planners ability to deviate from the path. However, this type of restriction suffers from naivety in regards to direction. This naivety is illustrated in the experiments in Figures 6 and 7. These figures demonstrate how a costmap can be modified to give preference to previously traversed routes. Throughout this paper, we refer to this path biasing strategy as the 'Map Restrictor'.

B. PaintPath

Standard costmaps are a scalar field representing the cost of passing through each point in the field. Usually this field is discretised as a two-dimensional or three-dimensional grid of values. These values determine a cost of traversing the grid cell and path planners then aim to minimise the integrated cost along the path length.

There can be, however, an ambiguity in how these costs are represented. A cost C in the range $[0, 1] \Rightarrow \{C \in: 0 \leq C \leq 1\}$ has the disadvantage that any obstacle, no matter how dangerous, can be traversed if the sum of costs along the path (including the obstacle) is less than the cost for all other paths. So instead, we represent the cost by a 0 to 1 'utility value', which is the reciprocal of cost. This allows obstacles and solid road lines, for example, to be absolutely untraversable (infinite cost), and it means terrain never has zero cost, so there is always a unique minimal cost path[1].

For many driving tasks, the direction of travel needs to be encoded in the map so that, for instance, the planned path does not guide the vehicle into oncoming traffic. This requires a vector-field costmap which in two-dimensions can be discretised into an image of vector-valued pixels.

In PaintPath the directional 'utility value' is calculated from the dot product $(\cdot)$ of the candidate vehicle direction vector with the per-pixel colour vector, resulting in a signed value in the range of $[-1, 1]$. However, as the utility value would desirably be in the range $[0, 1]$, the constant 1 is added to the dot product and the resulting value is divided by 2. An advantage of the dot product is that the resulting cost is linear on the input data. This is relevant as the colour map can be derived from multiple driving paths over the environment, and linearity ensures that such averaging of the colour vectors corresponds to an average of the cost of those vectors.

We use the HSV colour space, depicted in Figure 2, for the images, where hue H represents the intended driving direction, saturation S defines the directionality weight and value V represents the overall cost C of the cell. For a given candidate direction $\mathbf{d}$ the cost is

$$C = \frac{1}{V} + \frac{2}{S(1 + (\sin H, \cos H) \cdot \mathbf{d})} \quad (1)$$

[1]Strictly, there can be a finite number of minimal cost paths, but with vanishing probability.

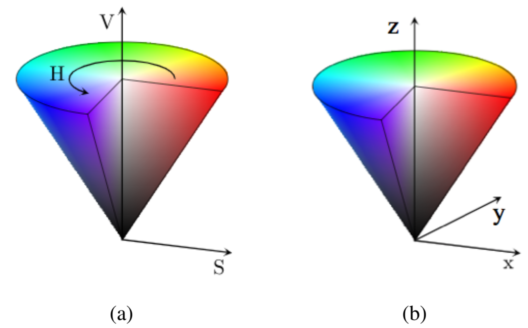


Fig. 2: Illustration of the HSV colour space. Figure (a) details the traditional representation with hue, saturation and value. Figure (b) relates this colour space to the variables used in the descriptions in this work.

This is a conical subset of a vector space, where black represents the highest cost (regardless of the direction of travel), and white represents the lowest cost. A bright colour represents the lowest cost when travelling in the direction corresponding to the colour's hue. This choice fits the driving task as the highest cost areas (obstacles) are high cost regardless of the approach direction, but the lowest cost areas (roads, open fields, and other vehicle operation areas) are the ones where directionality may be relevant.

In addition to loosely constraining the direction with the hue vector, we can more strictly constrain the drive direction using the value component V. As the environment is mapped, obstacles are marked as black (full cost), all remaining areas are marked as grey (medium cost). The driven trajectories are coloured according to the drive direction, and the colour interpolates to the default grey colour with distance from the trajectory path. This creates a set of softly bounded routes to constrain the driving direction.

In an omnidirectional area such as a compound or car park, the average of multiple drive passes over the same path in different directions creates an averaged white colour (Figure 3). This is brighter than the grey used as the non-obstacle default, which means that vehicles will still prefer to drive on the areas that were covered by vehicle traversals with no additional directional cost. It does not necessarily, however, forbid the vehicle to navigate to grey areas, as the system can be parameterised to only give less preference to grey, but not to avoid it completely in case a goal sits in one of that region.

C. Why use HSV?

Being a cone colour space, HSV allows direct mapping of directionality (Hue), directional certainty (Saturation) and obstacles (Value) in a format that is interpretable and editable by human operators. The addition of an obstacle layer allows the biasing of the planner to previously driven paths but does not restrict the planner to these paths. The use of not only direction, but also directional *certainty*, addresses the ambiguity of intersecting paths and bidirectional travel.

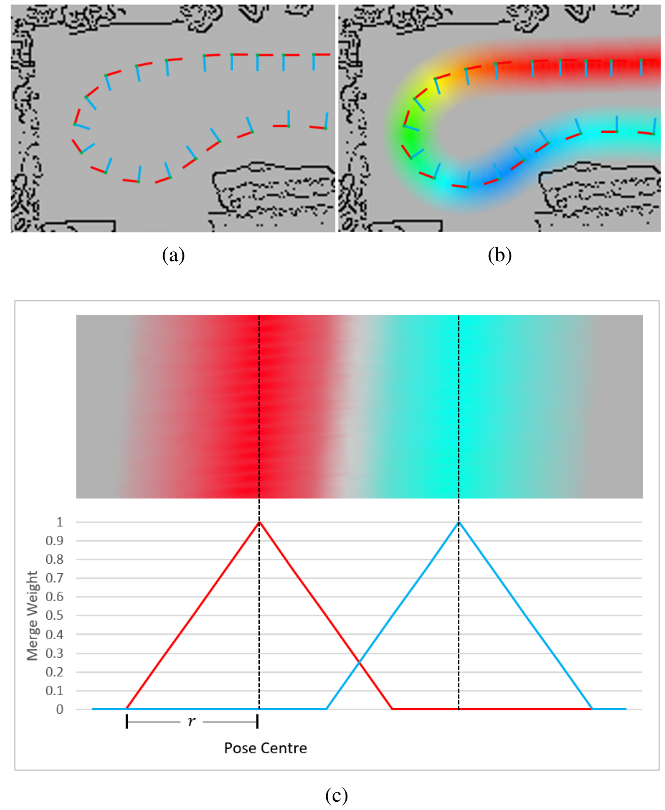


Fig. 3: In this figure (a) illustrates a sample vehicle trajectory with reference frames drawn at different poses. (b) shows how the directions in (a) relate to different colours. (c) details how the gradual colouring from the centre of the path to the background colour. The merge weight decreases from the pose centre to the maximum distance r.

D. Implementation

A simplified description of the implementation is shown in Figure 4. To provide the aforementioned smooth transitions from colour saturated paths to the default background colour and reduced saturation in bidirectional areas, two major processing steps must be completed: (i) the inflation of the trajectory and (ii) the averaging of colours. To create the smooth transition, a linearly decreasing 'merge weight' α ranging from 1 to 0 gradually reduces the saturation of the path. α is parameterised according to the maximum distance r from the centre of the trajectory $\mathcal{T}$, as illustrated in Figure 3. The direction of travel θ is extracted from $\mathcal{T}$ and used to calculate a colour representation vector $\boldsymbol{c}_n$ for each point within the distance r from the central pose. Let

$$\begin{aligned}
x &= \cos\theta \\
y &= \sin\theta \\
z &= 1 \\
\alpha_n &= 1 - \frac{\text{Distance From Pose}}{r} \\
\boldsymbol{c}_n &= (x, y, z)
\end{aligned} \tag{2}$$

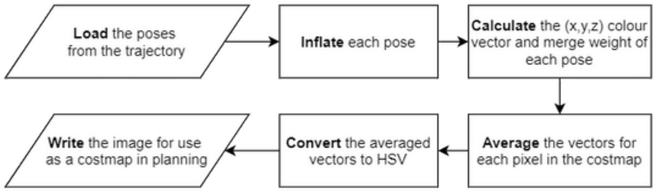


Fig. 4: Simplified description of trajectory processing. Poses are extracted from the trajectory before being inflated as described in Figure 3. For each point within this inflated pose, a colour vector $\boldsymbol{c}_n$ is calculated following the equations in (2). These vectors are subsequently averaged using equations (5) and (6). The resulting values for each pixel are converted to HSV (equation (7)), before being written out for use as a costmap in planning.

As there can be multiple vectors per pixel, an index n is assigned to each to distinguish between them. To effectively average the generated list of colours and produce desaturated colours in bidirectional areas, a weighted sum of the vectors generated from $\mathcal{T}$ and the background colour is computed. By averaging the x and y components of the colours rather than the direction, the saturation of the final colour can be reduced if the summed components act in opposite directions. In addition, by averaging the z components, a smooth transition to the background colour can be achieved. $\boldsymbol{c}_{\text{Background}}$ is defined as the vector containing the desired background colour (for example, if the desired background colour is grey, the vector $(0, 0, 0.7)$ could be used). Let

$$\omega = \max(\alpha_0, \alpha_1, \ldots, \alpha_N) \tag{3}$$

and

$$\omega_0 = \sum_{n=0}^{n=N} \alpha_n \tag{4}$$

where N is the value of the largest index for that pixel. Based on the equations above, an averaged vector $\boldsymbol{c}_{\text{Average}} = (x_{\text{Avg}}, y_{\text{Avg}}, z_{\text{Avg}})$ for each pixel in the image can be calculated. $\boldsymbol{c}_{\text{Average}}$ is defined as

$$\boldsymbol{c}_{\text{Average}} = \boldsymbol{c}_{\text{Sum}} \cdot \omega + \boldsymbol{c}_{\text{Background}} \cdot \max(0, 1 - \omega_0) \tag{5}$$

where

$$\boldsymbol{c}_{\text{Sum}} = \frac{\left(\sum_{n=0}^{n=N} \alpha_n \cdot \boldsymbol{c}_n\right) + \boldsymbol{c}_{\text{Background}} \cdot \max(0, 1 - \omega_0)}{\omega_0 + \max(0, 1 - \omega_0)} \tag{6}$$

This vector $\boldsymbol{c}_{\text{Average}}$ is then converted back to the HSV colour space according to

$$\begin{aligned} H &= \arctan \frac{y_{\text{Avg}}}{x_{\text{Avg}}} \\ S &= \sqrt{x_{\text{Avg}}^2 + y_{\text{Avg}}^2} \\ V &= z_{\text{Avg}} \end{aligned} \tag{7}$$

The resulting HSV values are incorporated into the costmap for use in path planning.

IV. Experiments

In this section we describe details of our hardware and software implementation followed by experimental results. We show a number of scenarios comparing PaintPath with standard costing approaches, illustrating the applicability of the method.

A. Hardware and Software Setup

Although our algorithm is not restricted to any specific type of localisation strategy, in our experiments we use LiDAR-intertial mapping, using the CSIRO SLAM algorithm [1], [2]. The sensors consists of a a Velodyne PUCK VLP-16 LiDAR and a Microstrain-CV5- IMU. The LiDAR assembly is additionally mounted on a spinning base, angled at $45°$ for increased point coverage [2] (Figure 5). The spinning base rotates at approximately $0.5\,\text{Hz}$ and LiDAR measurements are streamed in at $20\,\text{Hz}$.

In the autonomous vehicle experiment, a John Deere TE Electric Gator automated by CSIRO [29] was used as the test platform. The LiDAR mounted at a height of $1.88\,\text{m}$ above the robot frame, as illustrated in Figure 5.

The system was implemented in C++ using ROS [30], running on a LGA1151 CPU$2.8GHz$ and $64GB$ of RAM.

Fig. 5: Illustration of the test vehicle.

B. Results

Three main scenarios were chosen to evaluate and illustrate the differences in behaviour between PaintPath, Map Restrictor, and Traditional Costing. The scenarios include a variety of typical road networks, industrial areas, and off-road environments. Traditional Costing, in this case, refers to the free versus non-free binary costmap representation.

Scenario #1, depicted in Figure 6, shows a laned roadway with a goal point **G** placed approximately 50 meters away from the starting point **S**. The shortest path to the goal, however, goes in the opposite direction of travel according to site rules. When direction is not important, a short direct path is desirable as shown by the paths created by 'Traditional Costing' and Map Restrictor. If direction *is* important, the path taken with PaintPath, although significantly longer (as it performs a full-loop), obeys the encoded rules of directionality, always remaining on the left hand side of the road.

Scenario #2, shown in Figure 7, demonstrates similar results with both the paths taken by Map Restrictor and Traditional Costing violating traffic rules. In contrast, the route taken by PaintPath is significantly longer but adheres to these rules. The difference between Map Restrictor and Traditional Costing is highlighted in this scenario. The path taken by the Map Restrictor can be seen to be much smoother and closely follows the previously driven route, being a known plausible path.

Scenario #3, shown in Figure 8, consists of a very sloped off-road area. When considering how to traverse up a slope, the desired path is often indirect, following a shallower ascent. Scenario #3 (a) demonstrates both the paths taken by Map Restrictor and PaintPath following a safe route encoded by past experience, following a shallow gradient. In contrast the route taken by traditional costing directs the vehicle into the steepest section of the slope, placing the vehicle at risk of rolling. A secondary feature of off-road environments is that paths are often one directional. This characteristic is represented by both Map Restrictor and PaintPath again, allowing the vehicle to traverse down the slow and along a flat section of path rather than placing the vehicle at risk by moving along the slope.

Please note that in all figures showing trajectories in the experiments, the inverse colour is being displayed for visualisation purposes.

C. Computational Complexity

The computational complexity of the approach is split into two parts. Table I shows the computing times for generating the required costmaps for the different scenarios. Table II shows the times for computing the resulting trajectories. All the results were computed on a E5-2543 v3 CPU3.$4GHz$ and $126GB$ of RAM. It can be seen in Table I that the additional time utilised in the inclusion of a coloured path is dependent on the size of the map and the number of trajectory points. This time can be seen to vary from approximately $1.8 - 3.5$ times the time needed for an uncoloured map. Table II shows the time cost of performing additional costing functions in the A* algorithm. However, it is important to note that times cannot be directly compared, but must be considered alongside the trajectory length.

V. Conclusions

We have presented a novel and efficient method to incorporate directionality in costmaps. The PaintPath technique is applicable to map-based navigation, being particularly useful in areas which require autonomous navigation in a known map. The fundamental idea is to associate directionality in vehicle travel to directionality in the HSV colour space, using past travel direction to colourise the costmap, which, once colourised, is embedded with direction preference. We have presented a number of experiments showing situations in which PaintPath allows for proper path planning while standard binary costmaps would fail. Future work will extend the fundamental method to incorporate online adaptation and updating of the preferred path map as long-term navigation is performed after the initial map is created. Finally, the directionality concept can be extrapolated to include any other information from the trajectory into the map, such as velocity, acceleration, or any other arbitrary state or behaviour of the vehicles.

References

[1] M. Bosse and R. Zlot, "Continuous 3d scan-matching with a spinning 2d laser," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 2009, pp. 4312–4319.

[2] P. Egger, P. V. K. Borges, G. Catt, A. Pfrunder, R. Siegwart, and R. Dubé, "Posemap: Lifelong, multi-environment 3d lidar localization," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 3430–3437.

[3] K. S. Oberoi, G. DEL MONDO, Y. Dupuis, and P. Vasseur, "Spatial Modeling of Urban Road Traffic Using Graph Theory," in *Proceedings of Spatial Analysis and GEOmatics (SAGEO) 2017*. Rouen, France: INSA de rouen, Nov. 2017, pp. 264–277. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01643369

[4] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 33–55, March 2016.

[5] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 7 1968.

[6] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[7] S. M. LaValle, *Planning Algorithms*. USA: Cambridge University Press, 2006.

[8] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011. [Online]. Available: https://doi.org/10.1177/0278364911406761

[9] P. KrÃŒsi, P. Furgale, M. Bosse, and R. Siegwart, "Driving on point clouds: Motion planning, trajectory optimization, and terrain assessment in generic nonplanar environments," *Journal of Field Robotics*, vol. 34, no. 5, pp. 940–984, 2017. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21700

[10] G. Ishigami, M. Otsuki, and T. Kubota, "Range-dependent terrain mapping and multipath planning using cylindrical coordinates for a planetary exploration rover," *Journal of Field Robotics*, vol. 30, no. 4, pp. 536–551, 2013. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21462

[11] K. Iagnemma, F. Genot, and S. Dubowsky, "Rapid physics-based rough-terrain rover planning with sensor and control uncertainty," in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, vol. 3, May 1999, pp. 2286–2291 vol.3.

[12] I. Rekleitis, J.-L. Bedwani, E. Dupuis, T. Lamarche, and P. Allard, "Autonomous over-the-horizon navigation using lidar data," *Auton. Robots*, vol. 34, no. 1–2, p. 1–18, Jan. 2013. [Online]. Available: https://doi.org/10.1007/s10514-012-9309-9

[13] F. Ruetz, E. Hernández, M. Pfeiffer, H. Oleynikova, M. Cox, T. Lowe, and P. Borges, "Ovpc mesh: 3d free-space representation for local ground vehicle navigation," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8648–8654.

[14] A. Shum, K. Morris, and A. Khajepour, "Direction-dependent optimal path planning for autonomous vehicles," *Robotics and Autonomous Systems*, vol. 70, pp. 202 – 214, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889015000214

[15] I. Arvanitakis, A. Tzes, and M. Thanou, "Geodesic motion planning on 3d-terrains satisfying the robot's kinodynamic constraints," in *IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society*, Nov 2013, pp. 4144–4149.

[16] M. Pivtoraiko and A. Kelly, "Efficient constrained path planning via search in state lattices," in *Proceedings of The 8th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, September 2005.

[17] D. Gaw and A. Meystel, "Minimum-time navigation of an unmanned mobile robot in a 2-1/2d world with obstacles," in *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, vol. 3, April 1986, pp. 1670–1677.

Scenario #	Image Size	Number of Trajectories	Coloured Map (s)	Traditional Map (s)
1	4072×4886	7195	617.16	374.35
2	10358×14096	15510	2555.33	754.94
3 and 4	1954×2419	1142	107.53	36.4

TABLE I: Computing time taken to produce a coloured and traditional costmap seen in each test scenario. The image size is given in pixels.

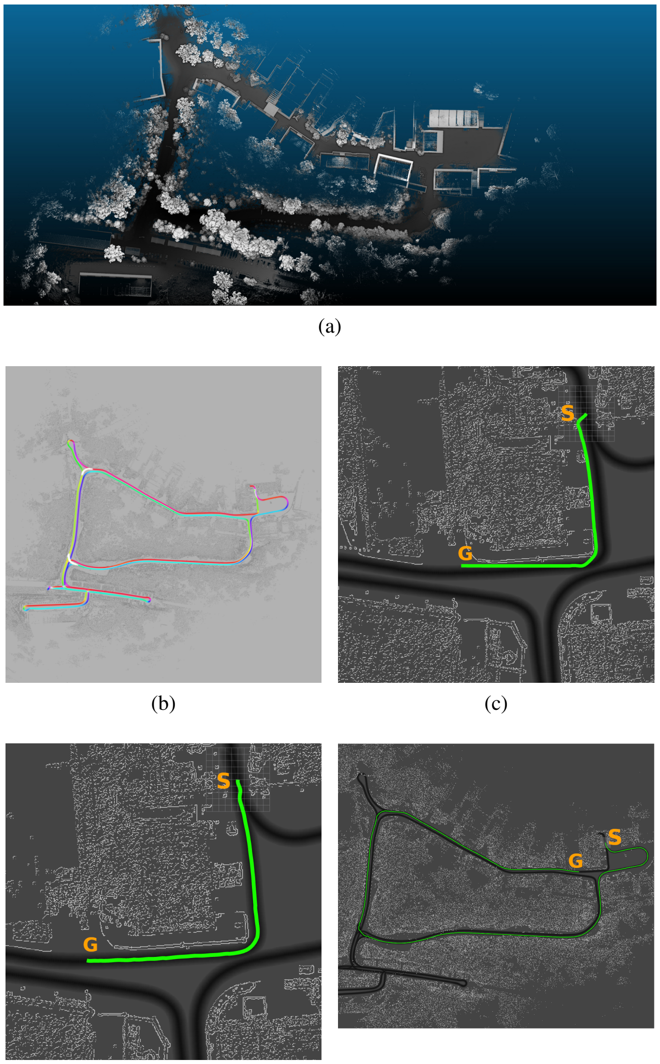


Fig. 6: Illustration of the varying paths generated in Scenario 1. Figure (a) shows the 3D pointcloud used to generate the map used in Scenario #1. The vertical height is represented by a grayscale gradient, black representing smaller values and white representing larger. Figure (b) shows the processed map used for planning, based on (a). Figure (c) through (e) show Traditional Costing, Map Restrictor and PaintPath, respectively, overlayed onto the value layer of the HSV image. The keys **S** and **G** represent the starting point and the end goal, respectively.

Scenario #	Traditional (s,m)	Map Restr. (s,m)	PaintPath (s,m)
1	0.03, 51.63	0.04, 42.40	6.81, 732.07
2	0.05, 115.58	0.15, 121.18	2.75, 335.73
3	0.03, 49.29	0.05, 53.51	0.21, 53.82
4	0.03, 49.72	0.05, 53.62	0.22, 53.25

TABLE II: The time taken by the path planner to compute the trajectories shown in each scenario discussed. Each entry includes the time taken to compute the trajectory in seconds (s), followed by the length of the trajectory in metres (m).

[18] Yi Guo, L. E. Parker, D. Jung, and Zhaoyang Dong, "Performance-based rough terrain navigation for nonholonomic mobile robots," in *IECON'03. 29th Annual Conference of the IEEE Industrial Electronics Society*, vol. 3, 2003, pp. 2811–2816 Vol.3.

[19] C. S. Swaminathan, T. P. Kucner, M. Magnusson, L. Palmieri, and A. J. Lilienthal, "Down the cliff: Flow-aware trajectory planning under motion pattern uncertainty," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 7403–7409.

[20] A. Howard and H. Seraji, "Vision-based terrain characterization and traversability assessment," *Journal of Robotic Systems*, vol. 18, no. 10, pp. 577–587, 2001.

[21] S. Karumanchi, T. Allen, T. Bailey, and S. Scheding, "Non-parametric learning to aid path planning over slopes," *The International Journal of Robotics Research*, vol. 29, no. 8, pp. 997–1018, 2010. [Online]. Available: https://doi.org/10.1177/0278364910370241

[22] A. Krebs, C. Pradalier, and R. Siegwart, "Adaptive rover behavior based on online empirical evaluation: Roverâterrain interaction and near-to-far learning," *Journal of Field Robotics*, vol. 27, no. 2, pp. 158–180, 2010.

[23] C. Rösmann, F. Hoffmann, and T. Bertram, "Kinodynamic trajectory optimization and control for car-like robots," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 5681–5686.

[24] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 4569–4574.

[25] D. A. Pomerleau, "Efficient training of artificial neural networks for autonomous navigation," *Neural Computation*, vol. 3, no. 1, pp. 88–97, March 1991.

[26] M. Riedmiller, M. Montemerlo, and H. Dahlkamp, "Learning to drive a real car in 20 minutes," in *2007 Frontiers in the Convergence of Bioscience and Information Technologies*, Oct 2007, pp. 645–650.

[27] B. D. Ziebart, A. L. Maas, A. K. Dey, and J. A. Bagnell, *Navigate like a Cabbie: Probabilistic Reasoning from Observed Context-Aware Behavior*. New York, NY, USA: Association for Computing Machinery, 2008, p. 322–331.

[28] J. Choi and K.-E. Kim, "Bayesian nonparametric feature construction for inverse reinforcement learning," in *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, ser. IJCAI '13. AAAI Press, 2013, p. 1287–1293.

[29] A. Pfrunder, P. V. 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)*. IEEE, 2017, pp. 2601–2608.

[30] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

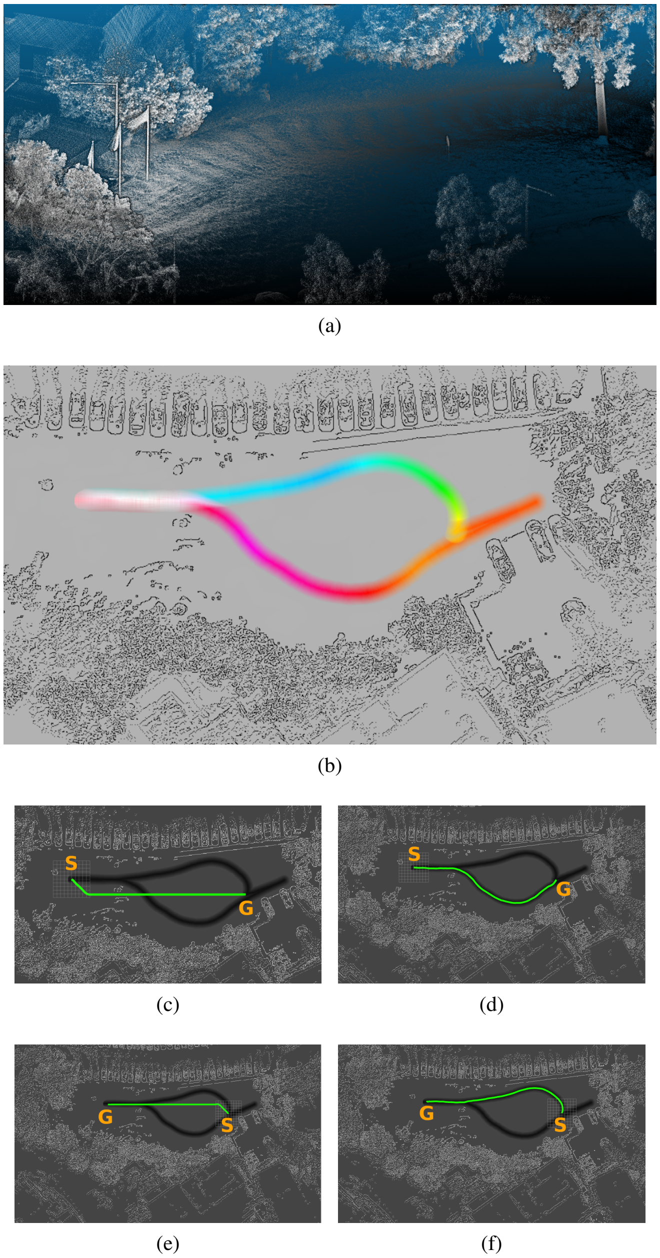


Fig. 8: Scenario #3 consists of very sloped terrain. Figure (a) shows the map used for planning. Figure (b) illustrates Traditional Costing and (c) shows the paths taken by both Map Restrictor and PaintPath. Both (b) and (c) show the path overlayed onto the value layer of the HSV image. The keys **S** and **G** represent the starting point and the end goal, respectively. In the opposite direction (i.e., **S** and **G** are swapped), the path taken by PaintPath is different and allows for a feasible route. The alternative path is illustrated in Figure (f), while the Traditional Costing in (e) is very similar to (b).

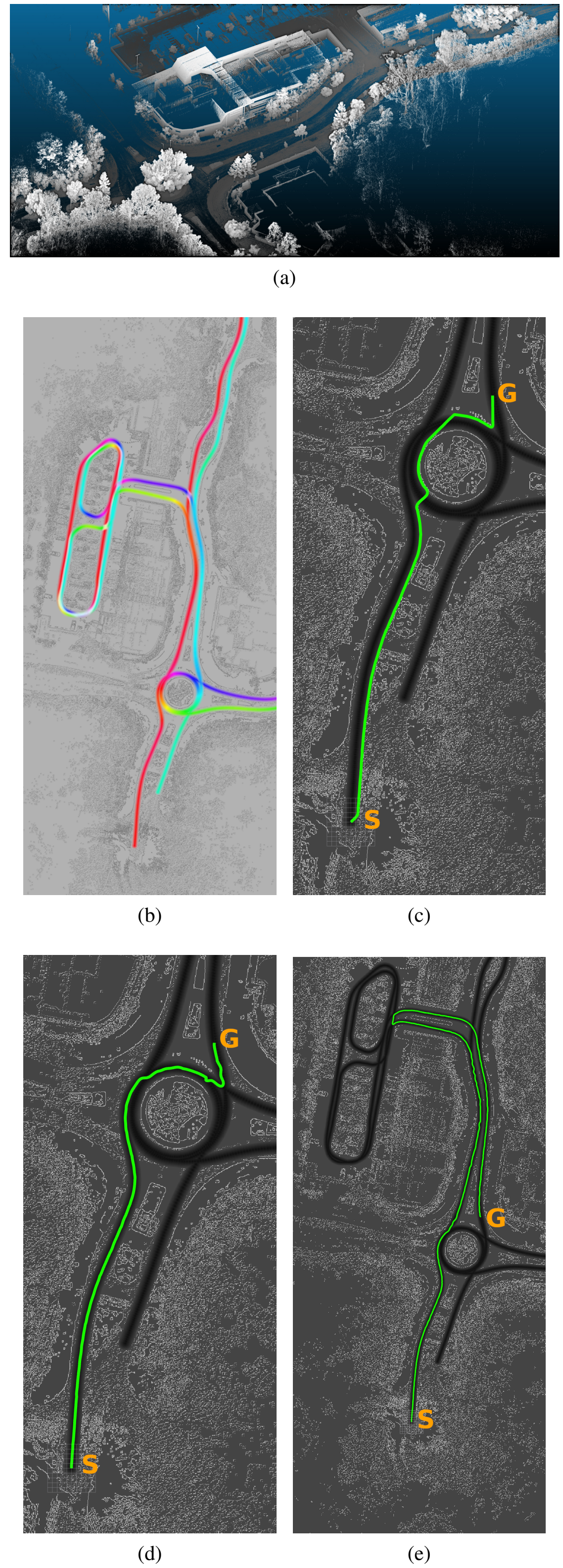


Fig. 7: Paths generated in Scenario 2. Figure (a) shows the map used for planning. Figure (b) through (d) show Traditional Costing, Map Restrictor and PaintPath, respectively, overlayed onto the value layer of the HSV image. The keys **S** and **G** represent the start and end goals.