Cinematographic and geometric criteria for virtual camera path generation for the visualization of shipwreck data

Katerine Davis[1], Vaibhav K. Viswanathan[2] Christopher M. Clark[2], Timothy Gambin[3] and Zoë J. Wood[1]

[1]*Cal Poly, San Luis Obispo*

[2]*Harvey Mudd College, Claremont*

[3]*University of Malta, Malta*

davis.kd.m@gmail.com, {vviswanathan, clark}@g.hmc.edu, timmy.gambin@um.edu.mt, zwood@calpoly.edu

Keywords: virtual camera path, probabilistic road map, rule of thirds, marine archeology visualization

Abstract: This paper presents the use of cinematographic and geometric measures for determining the path of a virtual camera for the generation of computer graphics video sequences focused on showing an underwater shipwreck. This work rises from the applied challenge of mapping underwater shipwrecks, reconstructing a computer graphics model then creating an educational visualization of the wreck. The primary algorithm presented in this work computes the optimal camera pitch and height along a path utilizing a probabilistic roadmap (PRM) that weights nodes using a computational models of cinematographic and geometric principles. These principles were used to evaluate potential viewpoints and influence whether or not a view is used in the final path. Specifically, the algorithm uses a computational evaluation of the cinematographic 'rule of thirds' and a geometric evaluation of the model normals relative to the camera viewpoint. A user study of video output from the system suggests that our computed paths are ranked higher than simple circular or random paths and that the 'rule of thirds' is a more important criteria than the geometric principle explored.

1 Introduction

Determining the path of a virtual camera through a digital scene in order to display the most relevant information in an aesthetically appealing way is an essential challenge when producing video output for a computer graphics applications. There exist millions of exquisite computer graphics models and scenes carefully crafted by artists and computer scientists. In order to show off these models and scenes, typically animators must hand create camera paths, often called fly-throughs. They do this by carefully choosing camera viewpoints that display specific features of the models and interpolating smooth paths between these points. This is a time consuming process that requires artistic ability to do well. We present a method to generate automatic fly-throughs for three-dimensional (3D) computer graphics scenes focused on a single model, building off the rich history of cinematographic principles, specifically focused on the general composition 'rule of thirds' (Krages, 2005). We use this cinematographic and an additional geometric consideration to generate videos that are appealing to users and feature good views of the most salient information in our scene.

In particular this work is focused on creating educational videos that allow a user to see and understand shipwrecks of archaeological significance. This work is a part of a larger project (von Fock et al., 2017) that uses underwater robots, sonar scans, video footage, and photogrammetry to create 3D computer graphics models of shipwrecks. Specifically, this work focuses on generating camera paths in a virtual underwater world, featuring the reconstruction of the World War II shipwreck, X127, mapped by the 2016 ICEX team.

Arising from this multi-disciplinary robotics and computer graphics project, our virtual camera work is inspired by the related problem of determining the path of the robot used in the same project. Specifically, to create a computer graphics reconstruction of the real world ship wrecks we use a photogrammetry pipeline that uses discrete frames from video data acquired by Autonomous Underwater Vehicle (AUV) deployments. To determine the path of the robot for optimal video coverage for the photogrammetry pipeline, we use a probabilistic roadmap (PRM) algorithm that utilizes a rapidly-exploring random tree to quickly cover a space and generate small maps with

good coverage. PRMs are ideal for classic robotics problems because they are highly customizable and can create paths that follow kinematic constraints of different robots. PRMs have also been used in a variety of other applications such as video games, character animation, and even computational biology (Hsu et al., 2003). For determining the path of the robot, a measure of information gain tied with the gradient of an occupancy map generated via sonar mapping is used to determine an optimal path for acquiring video footage of the wreck (Viswanathan et al., 2017). The video footage is then used in a photogrammetry pipeline to create a computer graphics model of the wreck.

Similarly, our algorithm to compute the path of the virtual camera uses a PRM planning algorithm to generate the fly-through camera path, given that PRMs have good coverage while keeping maps small. In order to evaluate a given node in our camera path, we use both cinematographic and geometric considerations to determine the value of the node. Specifically, these considerations evaluate viewpoints from heights and camera pitches generated along a specified path. The "rule of thirds" is used to characterize cinematographic principles, and object normals are representative of geometric evaluation.

This paper presents: our image processing system that renders a given 3D scene from a specified camera position and direction, then analyzes the image using computer vision with regards to cinematographic and geometric principles; our probabilistic roadmap algorithm that generates customizable, visually appealing camera paths for a fly-through of a 3D scene focused on a single model. We also present our evaluation of variations in the path selection criteria via user's evaluations of videos produced by our system. In general, users are pleased with the paths our system creates and prefer the paths generated with cinematographic principles over simple, flat paths.

2 Related Works

A large body of work exists related to probabilistic roadmaps, virtual cameras, and computational evaluation of image composition.

Cinematography and Image Composition A canonical work in the area of combining cinematography with computer science is that of Christianson et al. (Christianson et al., 1996). This paper created a camera control language to formalize cinematography principles for use in computer graphics. The idioms encoded focus on character interaction and movement rather than composition rules such as the rule of thirds used in our work.

The detection of the rule of thirds in still images is presented in (Mai et al., 2011). Mai et al. use saliency and generic objectness analysis to determine the subject of a photo and analyze where it is relative to the thirds lines. Other works detecting the rule of thirds utilize it along with other image composition rules to create aesthetically pleasing images through cropping and retargeting (Liu et al., 2010), (Setlur et al., 2005). Since our work focuses on scenes with a single main object we use a simple object detection technique to evaluate the rule of thirds.

Recently, evidence of the desirability of an automatic virtual camera planning, Unity Engine incorporated a virtual camera system, Cinemachine, to enable users with several virtual camera planning options (cin, 2017). The system includes the ability to specify a camera to track AI characters in a game, identify 'priority based shots' and other features.

Motion Planning Algorithms for Camera Control Recent work by Joubert, et. al., (Joubert et al., 2016), focuses on taking well-composed video footage from a quadcopter of human subjects interacting with each other using GPS and Inertia Measurement Unit sensors. To transition between static shots, the drone uses a real-time trajectory planning algorithm that tries to maintain visually pleasing shots while moving.

Another related work is that of Li and Cheng who navigate virtual environments with a real-time camera control module (Li and Cheng, 2008). This module generates camera motions that automatically control a third-person camera for use in video games. They maintain a real-time camera by using a lazy PRM that follows the game avatar. The PRM uses intercuts to avoid occlusion and collisions while preserving cinematography rules.

Other research related to automatic camera controls include visualizations focused on historical data (Stoev and Straer, 2002), examinations of viewpoint quality (Sokolov and Plemenos, 2005), and surveys of camera controls primarily applied to game settings (Christie and Olivier, 2009).

Probabilistic Roadmaps A vast body of work exists examining probabilistic roadmap theory and its applications in robotics (Li and Shie, 2002), including those using RRTs and PRMs for planning the path of an AUV, (Poppinga et al., 2011), (Lavalle, 1998), (Lavalle et al., 2000), (Candeloro et al., 2015).

The robotics portion of our shipwreck mapping project uses a PRM planner and an AUV to help create the shipwreck model (Viswanathan et al., 2017).

An occupancy map is created from a high level AUV scan of the shipwreck area and used in a PRM. The PRM creates additional AUV missions that make low altitude flyovers of the area optimized for "maximizing information gain" or getting good video data of the shipwreck. The photogrammetry pipeline (von Fock et al., 2017) used to create the model of the shipwreck requires photos of the shipwreck from as many angles as possible. Our "high weight node" list was inspired by this work's PRM that maximizes information gain. However, this work is optimizing for a high number of camera angles to use photogrammetry while we focus on maximizing viewpoints from a cinematographic and geometric perspective.

3 Algorithm

We present a system to generate fly-through animations using a robotics motion planning algorithm, probabilistic roadmaps. Cinematographic and geometric principles, specifically the rule of thirds and model normals, are used to generate virtual camera paths, resulting in visually pleasing video output. The sample scene used here features a digital 3D model of an actual shipwreck in Malta, the X127, placed in an underwater scene. The basic algorithm is as follows:

3.1 PRM Algorithm

PRMs use a RRT to explore a configuration space with small maps that provide good coverage. The configuration space used in this work is the height and pitch of the camera. At this time, the camera paths are generated along a set elliptical path with variation in radius. An example path shown from a top down view can be seen in Figure 1. The underwater world has various rock and seaweed models throughout and features a central shipwreck model that is our primary target for the camera to view from interesting angles.

Nodes are placed on a tree starting from a single root node, which is the camera's starting configuration. An entire tree of nodes is called a roadmap. This roadmap contains all nodes in one run of the PRM algorithm, regardless of whether or not these nodes end up on the final fly-through path. We use the term path to describe a set of nodes on our roadmap that form a complete course from start to end configuration of the desired length. Broadly, the algorithm adds nodes to a roadmap until a path of the desired node count is generated. First, a starting configuration is selected with a good viewpoint. Then, a node to expand from is selected using specific criteria and new nodes are

```
R(N, E) = Roadmap(Nodes, Edges);
Data: rootNode, pathLength
Result: Path of pathLength from rootNode
highWeightNodes[];
while currentPathLength < pathLength do
    // select node c to expand from
    if even iteration of loop then
        c ← highWeightNodes[iteration];
    end
    else
        do
            c ← rand node in R;
        while c < weightThreshold;
    end
    randomly generate c' from c;
    calculate edge e from c to c';
    R.add(c', e);
    currentPathLength ← c'.pathLength;
    c'.weight ← calculateWeight(c');
    if c'.weight > highWeightThreshold then
        highWeightNodes.add(c');
    end
end
```

Algorithm 1: High Level PRM Camera Path Generation Algorithm.

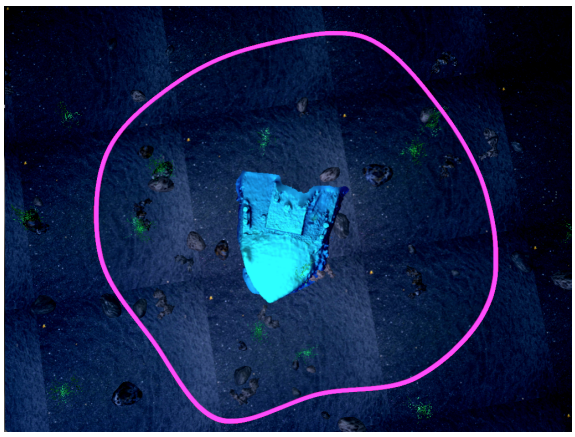


Figure 1: Top down view of the the shipwreck scene. Semi-elliptical path of varying radius shown in pink. Caustics turned off for clarity.

generated again with specific criteria discussed below. A node's value is calculated based on the measured quality of the rendered view from that node.

Root Node Generation All paths and roadmaps must begin with a starting configuration, specified in a root node. To start from a good initial configuration, 200 potential root nodes are generated by selecting random heights and pitches within reasonable start

values. The weight of all these nodes are calculated based on the current viewpoint evaluation criteria as described below in Section 3.2 and the best node is selected as the root node. Once the root node is chosen, it is added to the roadmap and the algorithm proceeds with the node selection step.

Node Selection For the algorithm to proceed, a node to expand from must be selected. We want to select a node with a better weight to generate more aesthetically pleasing paths. Throughout the entire path generation process, a list of "high weight" nodes is maintained that have values above a certain threshold. On half of the iterations through the path generation, a node from this list is selected to expand from. On the alternate loop iterations a random node with a weight above a threshold value in the roadmap is selected to expand from. This threshold is significantly lower than the threshold for the high weight node list. Additionally, this threshold decreases slightly each iteration in order that a node will be selected even if the roadmap only contains low quality nodes. Using node weight thresholds can be seen as a pruning step and this process helps keep the roadmap size small by not expanding the roadmap in directions with low quality nodes.

Node Generation In order to display our scene in a compelling way, the roadmap must expand out through our configuration space to show the shipwreck at a variety of angles. Once a node to expand from has been selected as described above, a new node is generated based off this old node. This node generation is done by taking the height of the old node and adding and subtracting a height delta. A random height is then chosen within this new range. The same process happens for pitch, selecting a random pitch within a delta of the old node's pitch. Height delta was typically in the range of 1-3 and pitch delta was generally between 5°and 15°. In our current virtual world, one meter in the real world is about 1.6 units in our virtual world.

Given that in general, fly-throughs follow a mostly circular path, we use a simplification on the planning of the camera and determine the x and z coordinates along with the yaw and roll by calculating the polar coordinates of a circle around the main model. We allow for variations in the radius (25-30 units) and eccentricity to create interesting paths. Figure 1, shows the top down view of one variation on a path generated by our system. In addition, during node generation, the height is constrained to be above the seafloor and below a specified maximum height. Once a new node is generated, we must evaluate the scene from that node's position and camera view direction.

3.2 Viewpoint Evaluation

The process of evaluating the camera viewpoint specified in a given node is based on cinematographic and geometric criteria. We choose to evaluate these criteria in the image domain, thus must render our computer graphics scene to evaluate the image. The node object is passed from the PRM generation code to the OpenGL code, rendered using OpenGL with the results then evaluated using OpenCV. The calculated weight is then set for the node and used in the node selection process for the next iteration of the PRM loop.

The 3D underwater OpenGL scene includes sand, rocks, seaweed, bubbles, caustics, and the X127 shipwreck model. An example of all these elements in the scene can be seen in Figure 5. The primary element in the scene is the X127 shipwreck, created from an actual shipwreck in Malta using a photogrammetry pipeline using input video gathered from an AUV. Since the X127 wreck lays on a diagonal, the deeper part of the wreck sits in water that is too murky for the AUV to take high quality video footage.

A PRM node's position and direction vectors are used to set a virtual camera in the OpenGL scene. Then the resulting scene is written to a framebuffer and transferred to the OpenCV image format, which is used to evaluate how closely the image follows the rule of thirds and whether the model is primarily facing towards the camera.

Rule of Thirds Detection One of the canonical rules of aesthetics from a cinematographic perspective is the rule of thirds, which can be described as splitting an image up into three uniform sized sections, both horizontally and vertically. The subject of the photo or horizon line should be placed along one of these lines or at the intersection of two such lines as can be seen in many famous photographs and paintings. Thus for automatic camera path generation a good view of the scene should follow the rule of thirds so our algorithm evaluates nodes relative to how well a given view follows this rule.

We combine several of the methods provided in OpenCV to evaluate how well an image conforms to the rule of thirds. First the input image is converted to grayscale and blurred. The image is then thresholded to detect edges and then the contours or outlines of the shapes in the image are computed. The contours are approximated as polygons and bounding rectangles and circles are formed. An example of the output from these computations can be seen in Figure 2.

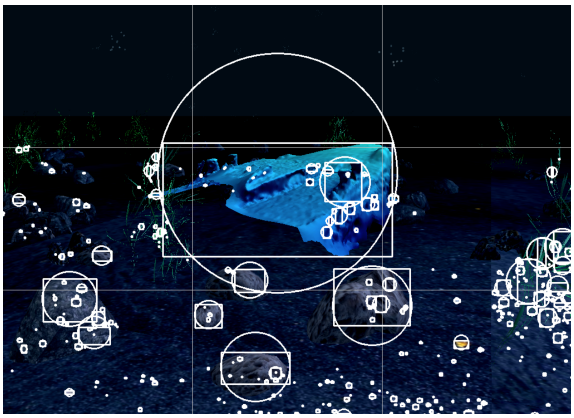


Figure 2: OpenCV methodology to draw bounding rectangles and circles around detected contours in an image.

To use this methodology in our program, a kernel size of five by five is used in the normalized box filter. In addition, bounding circles are eliminated and only one bounding rectangle is preserved, specifically the largest. There are various ways to consider a rule of thirds composition, thus we conducted a user study with static images of our scene and found that users preferred the top horizontal line of the ship to be aligned with top third horizontal extent of the image, i.e. where the shipwreck created a horizon line at the top thirds line of the image. Thus, the largest y coordinate of the largest bounding rectangle is found and used to compare against the top third of the entire image to evaluate whether the key feature of our scene conforms to the rule of thirds as shown in Figure 3. See Equation 1. This measure is set as the weight of the node to be used in node selection. A limitation of our method is that we currently can only have one main model in our scene for our detection algorithm to work well.

$$weight = |(BBox.y/ImageHeight - 2/3)| \quad (1)$$

Model Normals Detection From a geometric perspective, we want camera views that show us as much of the model as possible. The node weight calculation detects how much of a given object is pointing towards the camera via use of the model's normals. Specifically, a vertex shader is used to transform the normals into the camera's eye space. When these normals are then used to color the rasterized fragments, a normal that is pointing primarily in the direction of the camera will be colored blue. OpenCV is used to detect the proportion of pixels that represent fragments facing the camera, which is then used to weight the nodes in the search space.

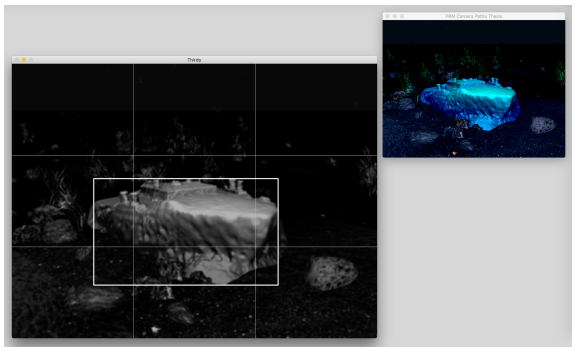


Figure 3: Rule of thirds detection. On the right the OpenGL scene is rendered and on the left, OpenCV is used to detect the largest shape in the image and compared to the thirds lines of the image.

Thirds and Normals Combination The rule of thirds and normal detection both have shortcomings so we explored combining them to try to get the best path possible. The rule of thirds is optimal for creating smooth paths but can occasionally score well for views of the wreck from poor angles. For example, a very low viewpoint, almost from the seafloor can create a path following the rule of thirds just as well as one from a higher, more scenic view. The normal detection does a much better job with this, but can create very jumpy paths that leap from one extreme angle to another. To combine the thirds scores that are better if they are lower and normals scores where a higher score is better, we use the following equation:

$$weight = (1 - |(BBox.y/ImageHeight - 2/3)|) + (\sum_{n=0}^{ImageSize} pixel.blue > threshold)/ImageSize \quad (2)$$

An example of a node viewpoint being evaluated with both methods is shown in Figure 4. This combination method was noticeably slower because it required rendering the scene, converting the image to OpenCV format, and running OpenCV algorithms twice for each node viewpoint.

Node Weight & Best Weight List To drive the path towards a reasonable solution in a shorter amount of time, a highWeightNode list was used during the node selection process and every other iteration of the PRM loop, a node is selected from this list. For the rule of thirds, nodes with weight 0.04 or lower were added to the best weight list while nodes with weight 0.25 or higher were considered good for normals scores. Since these methods were combined by subtracting the thirds score from 1 to create positive values, a good combination score was defined as 1.21.

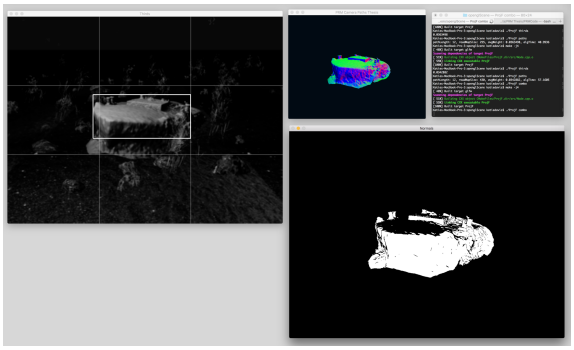


Figure 4: Combination node weight calculation. Rule of thirds and model normals are detected for each viewpoint and the combined weights are set as the node weight.

Storing Complete Paths Each time a new node is added to the roadmap, the algorithm checks if that node completes a path of the desired path length. If a full length path is found then the roadmap is traversed from the newest node back up to the root node. Position and direction vectors of each node in the path are then written out to a file allowing our system to playback a path from a file. Videos can then be created by using the full list of camera positions and directions to create a Catmull-Rom spline between the nodes. A spline path interpolating between nodes in a generated path can be seen in pink in Figure 1.

4 Results

In this work, a PRM algorithm adds nodes to a tree until a path of sufficient node count has been generated. A node to expand from is selected, then new nodes are generated given precise parameters. Virtual camera viewpoints, specified in new nodes, are evaluated on cinematographic and geometric principles, i.e., the rule of thirds and model normals, and assigned a weight. These weights are used in the node selection process, generating aesthetically pleasing camera paths.

Parameters to control the path generation include variations on the path such as length, radius, elliptical factor, radius variation, camera speed (via spline computation between nodes) and camera specific variables such as initial height and pitch, height and pitch delta, and finally evaluation criteria such as color and blur threshold factors. PRM variables include "high weight" list threshold for thirds, normals, and combination, frequency of selecting from high weight list versus random selection, threshold when selecting a random node to expand from for thirds, normals, and combination, and limits when changing the threshold for selecting a random node.

Table 1: Viewpoint evaluation method study results. 1 is most preferred and 5 is least preferred. Columns 2 - 6 show the number of participants that gave that ranking and the last column shows the average score for each video. Video A was the highest ranked with an average score of 2.14.

Video	1	2	3	4	5	Avg.
A (Thirds)	13	10	8	2	2	2.14
B (Random)	0	0	2	11	22	4.57
C (Combo)	10	5	9	8	3	2.69
D (Flat)	7	11	10	5	2	2.54
E (Normals)	5	9	6	9	6	3.06

4.1 User Studies

With so many different parameters we evaluated our results with a user study comparing output videos. Output videos include ones that keep all parameters the same and vary the method used for calculating node weights (the rule of thirds, model normals, combination), judged against a completely flat path, and a completely random path. A second study included videos that used a combination of thirds and normals to calculate node weights but then varies path parameters, including path length, radius, deltas, elliptical factor of path, and more. The camera viewpoints in the best generated path from both studies can be seen in Figure 5.

In each survey, participants were asked to rank five 15 second videos in order of preference.

Viewpoint Evaluation Methods Study The first study focused on evaluating camera paths that were generated via different evaluation methods. For this survey, path parameters were kept constant and only node evaluation criteria were changed. The paths had a slightly elliptical orbit that was 0.75 width in the z component. The initial height varied between 4 and 12 and the initial pitch angle varied between 45° and 0° below the horizon line. The max height delta possible was 2.5 and the max pitch change was 7.5°. The thresholds of what defined a "good" node were kept constant throughout all path generation for both surveys. The viewpoint evaluation methods used to produce each video are as follows: **Video D** was a completely flat path with no variation in height, **Video B** displayed a path that was randomly generated with the above parameters, **Videos A, E, and C** used the rule of thirds, model normals, and a combination of both respectively to evaluate potential camera viewpoints.

To avoid bias, 30 paths were generated of each type and the path with the best average node score was chosen for the study. Results of the study for the 35 participants are displayed in Table 1.

The results show that people preferred the path

generated with only the rule of thirds. The second highest path was flat, followed closely by the combination path. Normals came in a solid fourth with random at a strong last place. The results were surprisingly spread out, with all paths except random having at least two rankings for every possible position.

Path Types Study This study kept the viewpoint evaluation method constant, using a combination of rule of thirds and model normals. The variable being studied here was path type. Videos A through E respectively displayed paths with large variation, many nodes, small variation and perfectly circular, largely elliptical, and random. Excluding random, 30 paths of each type were generated and the best scoring path of each type was used in the study. Since the random path used no evaluation criteria, one random path was generated and used in the study. This study had 32 participants total. The primary conclusion from this study was that user's preferred the automatic camera paths and in particular the paths with the largest allowable variations in view and path radius variations, with 18 of 32 people ranking it number 1 and 12 people ranking it number 2.

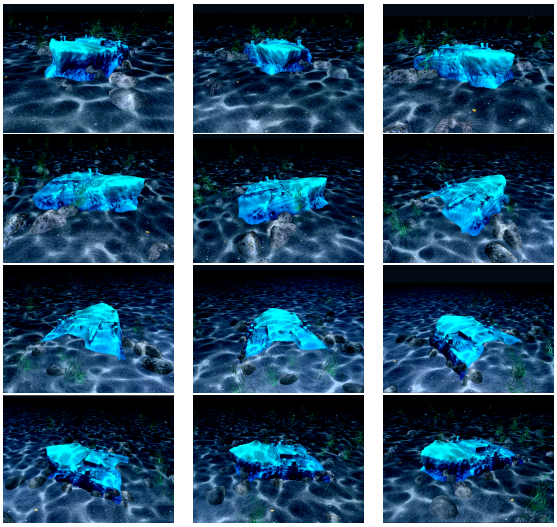


Figure 5: Camera viewpoints used in the winning path in the user studies. A spline interpolates between these views to create a 15 second fly-through of the scene.

From these studies, we can conclude that overall the use of our automatic camera path generation is appealing to users, winning out over planar circular paths and random paths and with the rule of thirds being more important than our current geometric criteria of using normals alone. There is a large variety of variations and future work discussed below.

Table 2: Information corresponding to best paths; average weight of all nodes in final path, number of nodes in the entire roadmap and seconds spent generating the roadmap.

Criteria	Num nodes	Avg Weight	Roadmap Size	Time
Thirds	12	0.0131	1134	38.5
Norms.	12	0.1681	202	6.4
both	12	1.0467	663	36.4
both	12	0.99642	243	19.5
both	45	0.9858	1080	48.8
both	10	0.993663	56	12.4
both	12	0.917036	238	19.4

4.2 Performance and limitations

Implementation of this work was done in C++ using the Open Graphics Library application programming interface and the Open Computer Vision library. It was run on a 2015 MacBook Pro running macOS Sierra 10.12.3 with a 2.9 GHz Intel Core i5 processor.

All motion planning algorithms can be time and memory consuming to run, with the PRM being no exception. One way to generate a smooth camera path is to put so many nodes on a circular path that each viewpoint is extremely close together. The camera will flow smoothly through node transitions due to the short distance between viewpoints. We attempted this in our initial implementation, generating paths of length 40 in about 10 seconds with full roadmaps containing over 20,000,000 nodes. The time it took to generate longer paths went up exponentially, taking 30 seconds to generate a path of length 50 and a path of length 60 could not be generated in five minutes. We experimented with many different optimizations such as limiting the number of children per node, spatially hashing the configuration space and limiting the number of nodes in each spatial range, and selecting a random point in the configuration space and expanding off the nearest node to it.

Using splines to interpolate the camera path between viewpoints, enabled us to use fewer nodes in a path, which in turn meant lower time and memory costs, allowing us to generate and compare a larger number of paths. The final algorithm uses around 12 nodes to generate a path. This decision is justified by the assumption that a movie director creating an overview scene like our circular fly-through would only place a few key viewpoints and let the camera operator move smoothly between these views. The evaluation type, path length, average node weight, roadmap size, and time to generate the path for the paths evaluated in our user study, excluding the flat and random ones, are summarized in Table 2.

Limitations and future work Our fly-through generation system has a number of limitations that will be improved in future iterations of this project. First, our system currently only works on scenes with one central model because our rule of thirds detection algorithm only looks at the single largest shape. A more complete comparison with existing work would also be highly valuable for positioning this work in a larger context. We are exploring adding more motion modeling constraints on the camera motion in terms of pitch and height and adding other variables that can be varied and computed using the PRM. These variables notably include completely free positioning of the camera and solving for the yaw of the camera. We are also exploring additional evaluation criteria such as object detection of specific items of interest to the viewer and including camera collision detection.

ACKNOWLEDGEMENTS

We acknowledge the hard work on the entire ICEX 2016 team.

REFERENCES

(2017). Cinemachine camera tools in unity engine. https://blogs.unity3d.com/2017/01/23/cinemachine-base-rig-now-available-for-free-creator-adam-myhill-joins-unity//.

Candeloro, M., Mosciaro, F., Srensen, A. J., Ippoliti, G., and Ludvigsen, M. (2015). Sensor-based autonomous path-planner for sea-bottom exploration and mosaicking. In *IFAC Conference on Manoeuvring and Control of Marine Craft*, pages 31–36.

Christianson, D. B., Anderson, S. E., and wei He, L. (1996). Declarative camera control for automatic cinematography. *American Association for Artificial Intelligence*.

Christie, M. and Olivier, P. (2009). Camera control in computer graphics: Models, techniques and applications. *ACM Siggraph Asia Courses*.

Hsu, D., Jiang, T., Reif, J., and Sun, Z. (2003). The bridge test for sampling narrow passages with probabilistic roadmap planners. In *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, volume 3, pages 4420–4426 vol.3.

Joubert, N., E, J. L., Goldman, D. B., Berthouzoz, F., Roberts, M., Landay, J. A., and Hanrahan, P. (2016). Towards a Drone Cinematographer: Guiding Quadrotor Cameras using Visual Composition Principles. *ArXiv e-prints*.

Krages, B. P. (2005). *Photography: The Art of Composition*. Allworth Press.

Lavalle, S. M. (1998). Rapidly-exploring random trees: A new tool for path planning. Technical report.

Lavalle, S. M., Kuffner, J. J., and Jr. (2000). Rapidly-exploring random trees: Progress and prospects. In *Algorithmic and Computational Robotics: New Directions*, pages 293–308.

Li, T.-Y. and Cheng, C.-C. (2008). *Real-Time Camera Planning for Navigation in Virtual Environments*, pages 118–129. Springer Berlin Heidelberg, Berlin, Heidelberg.

Li, T.-Y. and Shie, Y.-C. (2002). An incremental learning approach to motion planning with roadmap management. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, volume 4, pages 3411–3416 vol.4.

Liu, L., Chen, R., Wolf, L., and Cohen-Or, D. (2010). Optimizing photo composition. *Computer Graphics Forum*, 29(2):469–478.

Mai, L., Le, H., Niu, Y., and Liu, F. (2011). Rule of thirds detection from photograph. In *Proceedings of the 2011 IEEE International Symposium on Multimedia*, ISM '11, pages 91–96, Washington, DC, USA. IEEE Computer Society.

Poppinga, J., Birk, A., Pathak, K., and Vaskevicius, N. (2011). Fast 6-dof path planning for autonomous underwater vehicles (auv) based on 3d plane mapping. In *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 1–6. IEEE Press, IEEE Press.

Setlur, V., Takagi, S., Raskar, R., Gleicher, M., and Gooch, B. (2005). Automatic image retargeting. In *Proceedings of the 4th International Conference on Mobile and Ubiquitous Multimedia*, MUM '05, pages 59–68, New York, NY, USA. ACM.

Sokolov, D. and Plemenos, D. (2005). Viewpoint quality and scene understanding. *Proceedings of the International Conference on Virtual Reality, Archaeology and Intelligent Cultural Heritage*.

Stoev, S. L. and Straer, W. (2002). A case study on automatic camera placement and motion for visualizing historical data.

Viswanathan, V. K., Lobo, Z., Lupanow, J., von Fock, S. M. T. S., Wood, Z., Gambin, T., and Clark, C. (2017). Auv motion-planning for photogrammetric reconstruction of marine archaeological sites. In *Proceedings of the 2017 IEEE International Conference on Robotics and Automation*.

von Fock, S. M. T. S., Bilich, S., Davis, K., Viswanthan, V. K., Lobo, Z., Lupanow, J., Clark, C., Gambin, T., and Wood, Z. (2017). Pipeline for reconstruction and visualization of underwater archaeology sites using photogrammetry. In *Proceedings of the 2017 ISCA International Conference on Computers and Their Applications*.