Autonomous Robot Navigation and Localization on 3D Mesh Surfaces in ROS

Sebastian Pütz, Alexander Mock

MESH
NAV

Mesh Navigation runs with Move Base Flex

Navigation Control

- Universal & Flexible Robot Navigation Control System:

Move Base Flex (MBF)

github.com/magazino/move_base_flex

MBF
MOVE BASE FLEX

Sebastian Pütz and Jorge Santos Simón, *Introducing a Highly Flexible Navigation Framework: Move Base Flex presented at the* ***ROSCon 2017, Vancouver***, Canada, Sep. 22, 2017, https://vimeo.com/236174072

Path Planning

- Navigation & Path Planning for Universal Map Representation:

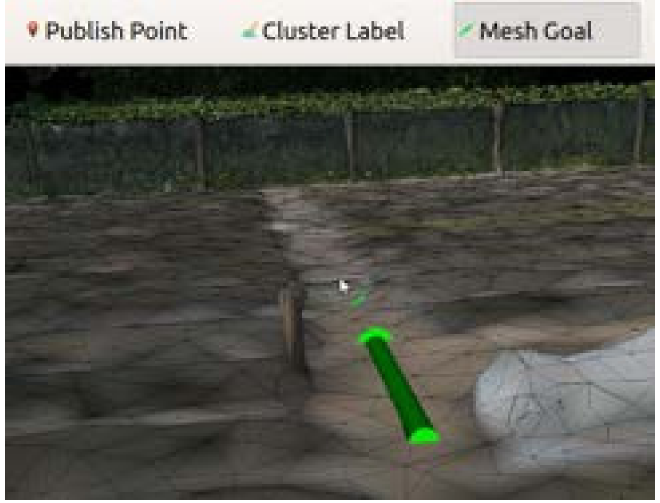
Mesh Navigation (MeshNav)

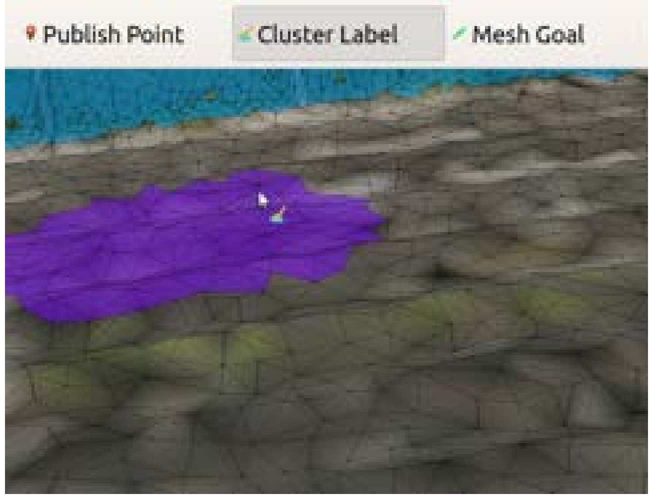
github.com/naturerobots/mesh_navigation

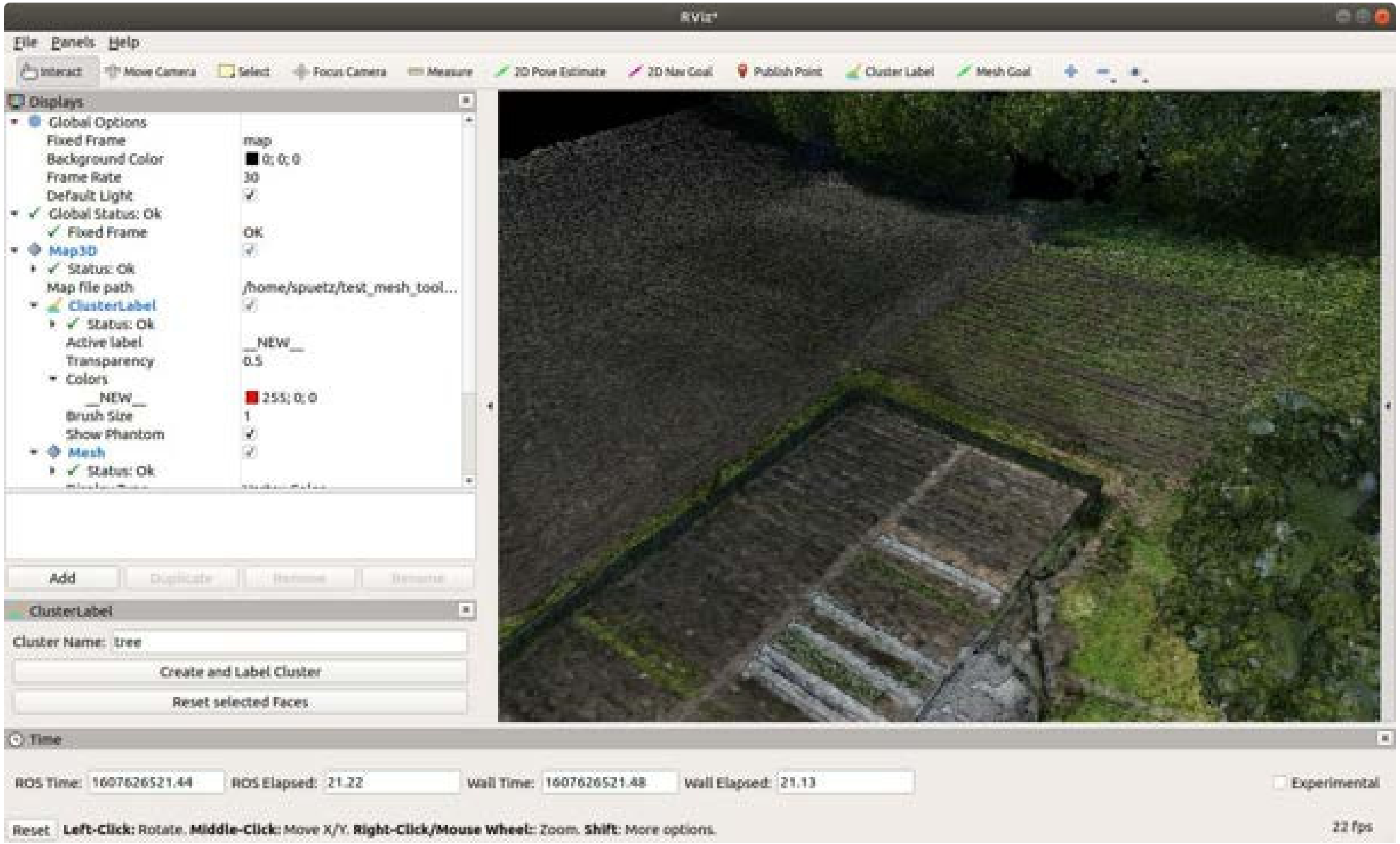
MESH
NAV

Sebastian Pütz (2022), *Navigation Control & Path Planning for Autonomous Mobile Robots,* Doctoral dissertation, Osnabrück University, https://doi.org/10.48693/69

Mesh Tools







Sebastian Pütz, Introducing Tools for Storing, Rendering and Annotating Triangle Meshes in ROS and RViz, presented at the **ROSCon 2018, Madrid**, Spain, Sep. 30, 2018, Available: https://vimeo.com/293617680

Sebastian Pütz, Thomas Wiemann, and Joachim Hertzberg, Tools for Visualizing, Annotating and Storing Triangle Meshes in ROS and RViz, in Proc. 9th European Conference on Mobile Robotics (ECMR 2019), Prague, Czech Republic, Sep. 2019

Real-World Datasets

Farmer’s Pit Stemwede

Stone Quarry Brockum

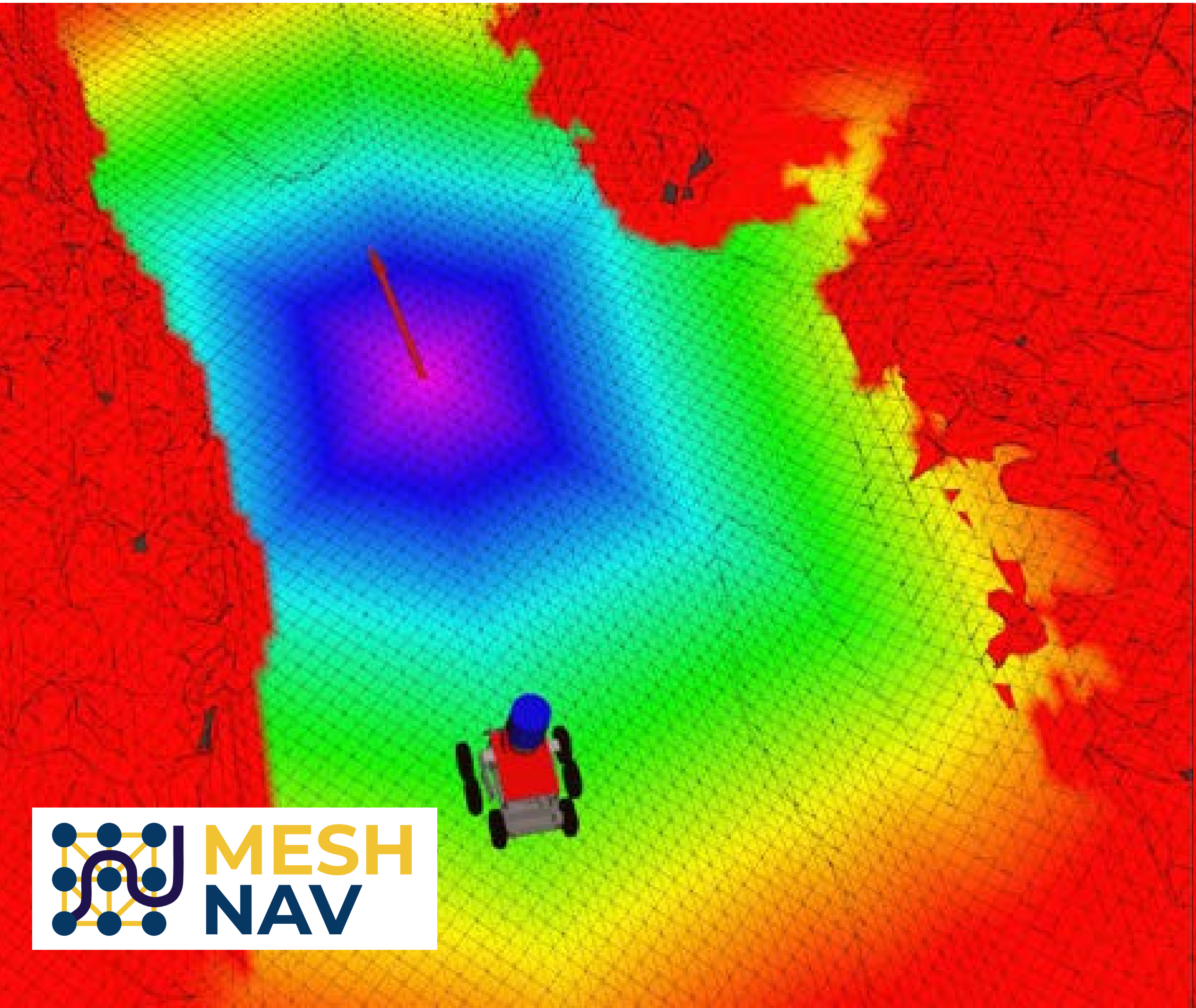
Markt Garden Ibbenbueren

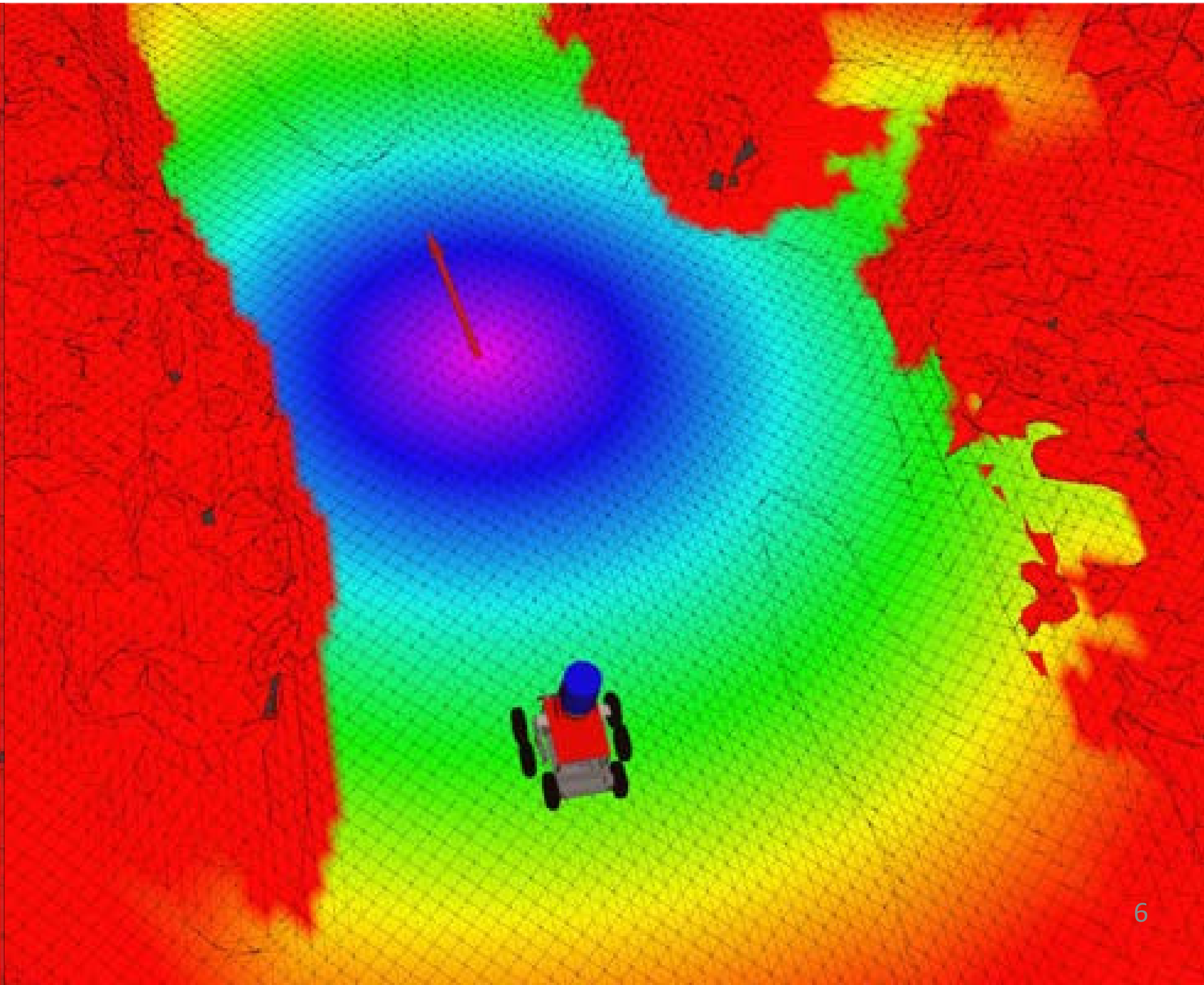
Physics Campus Westerberg

Dijkstra's Algorithm

vs.

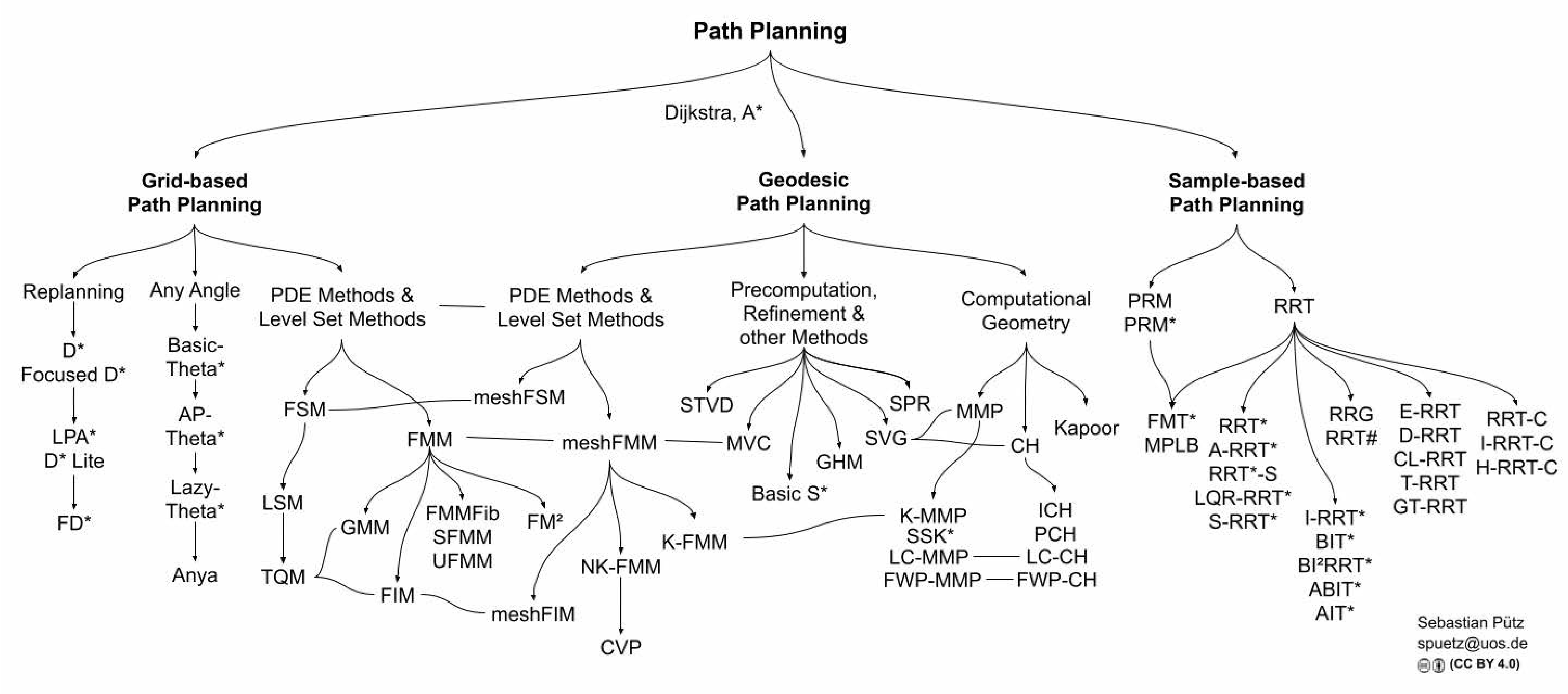
Continuous Vector Field Planner (CVP)





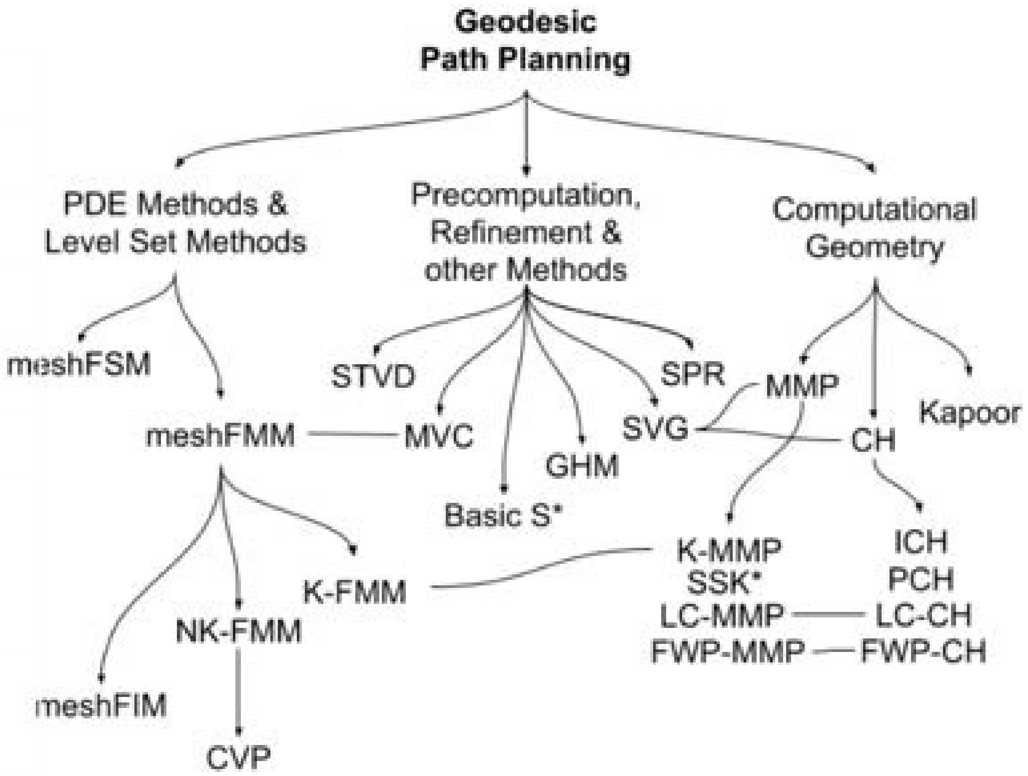
MESH NAV

Path Planning Algorithms



Significant Geodesic Path Planning Evolutions

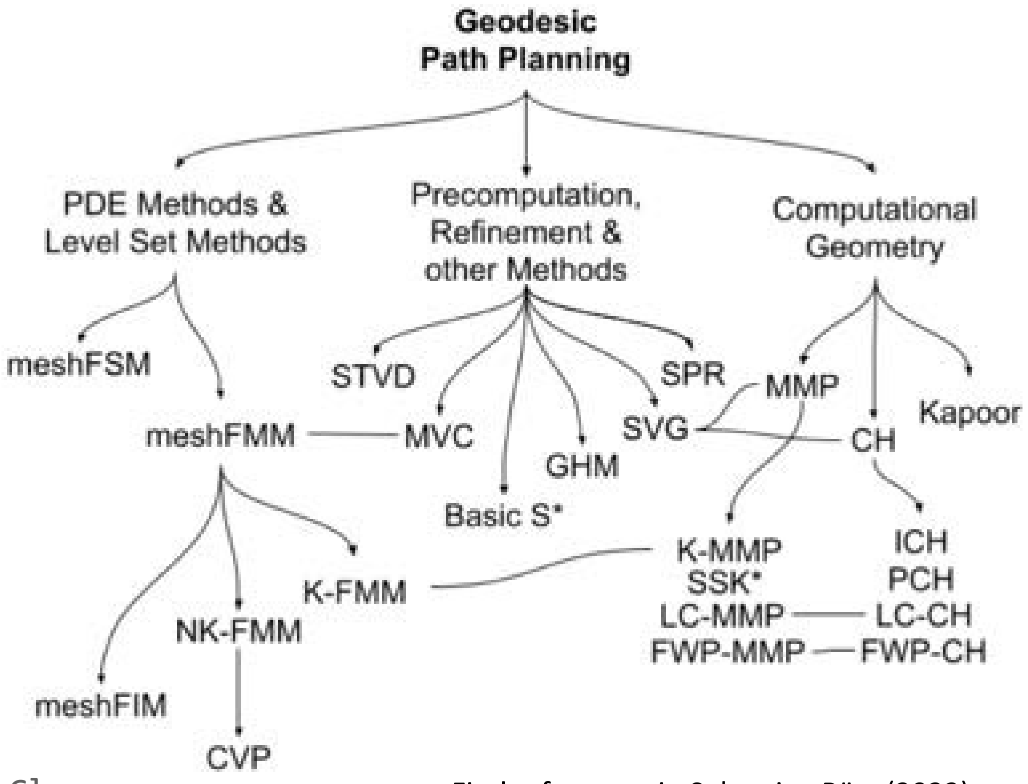
Year	Approach	Asym. runtime
1985	O'Rourke et al. [218]	$O(n^5)$
1986	Sharir and Schorr [276]	$O(n^3 \log n)$
1987	Mitchell et al. (**MMP**) [199]	$O(n^2 \log n)$
1990	Chen and Han. (**CH**) [38]	$O(n^2)$
1999	Kapoor [138]	$O(n \log^2 n)$
1997	Lanthier et al. [171, 172], and in 2000 Kanai and Suzuki [136] Steiner Points Refinement (SPR)	
1998	Kimmel and Sethian [152] (**meshFMM**)	$O(n \log n)$



Find references in Sebastian Pütz (2022) *Navigation Control & Path Planning for Autonomous Mobile Robots,* Doctoral dissertation, Osnabrück University, https://doi.org/10.48693/69

Modern Geodesic Path Planning Algorithms

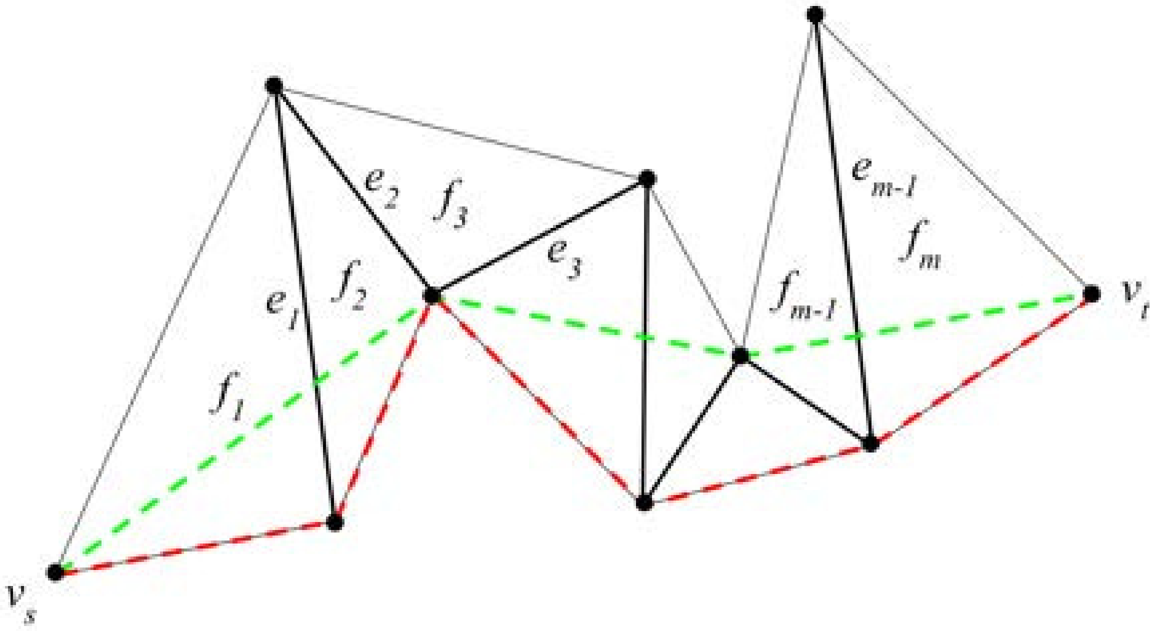
- 2002: Single Source FMM by Novotni and Klein (**NK-FMM**) [210]
- 2004: Flat-Exact based on FMM by Kirsanov (**K-FMM**) [153, 154]
- 2004: Improved exact MMP by Kirsanov (**K-MMP**) [153, 154]
- 2005: Fast and improved MMP by Surazhsky et al. (**SSK***) [290]
- 2007: Fast Sweeping Method (**meshFIM**) by Qian et al. [236]
- 2009: Improved CH (**ICH**) by Xin and Wang [313]
- 2011: Fast Iterative Method (**meshFIM**) by Fu et al. [90]
- 2013: Saddle Vertex Graph (**SVG**) by Ying et al. [320]
- 2013: Geodesics Heat Method (**GHM**) by Crane et al. [51]
- 2013: Short-Term Vector Dijkstra (**STVD**) by Campen et al. [35]
- 2014: Parallel CH (**PCH**) by Ying et al. [321]
- 2015: Fast Wavefront Propagation CH (**FWP-CH**) by Xu et al. [316]
- 2015: Fast Wavefront Propagation MMP (**FWP-MMP**) by Xu et al. [316]
- 2019: **Basic S*** by Bhattacharya [14]
- 2021: Continous Vector Field Planner (**CVP**) by Sebastian Pütz [235]



Find references in Sebastian Pütz (2022) *Navigation Control & Path Planning for Autonomous Mobile Robots,* Doctoral dissertation, Osnabrück University, https://doi.org/10.48693/69

Geodesic Path Planning

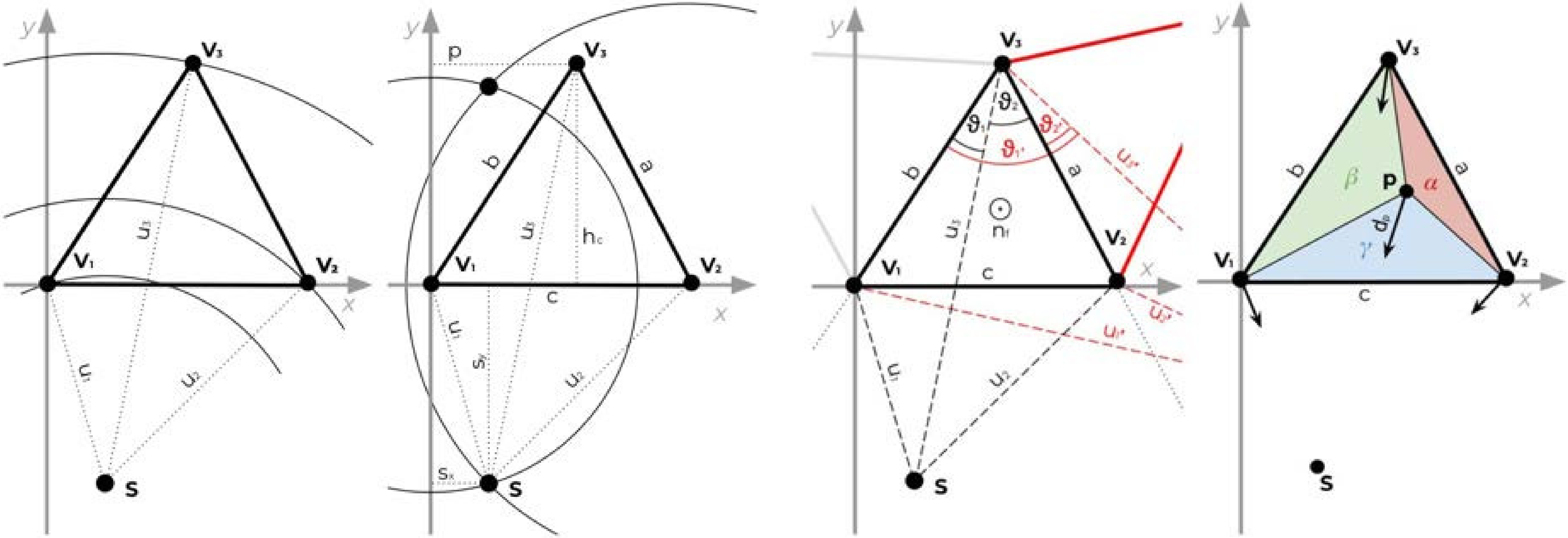
- Used in robotics, geographic, information systems, medical imaging, flight simulation, and water flow analysis, CAD
- Rarely applied and implemented in terms of robot navigation, not to mention applications for real outdoor environments
- Dijkstra's algorithm can lead to suboptimal paths on triangular meshes
- Many geodesic algorithms are similar to Dijkstra's algorithm and solve the SSSP problem, thus a comparison to Dijkstra's algorithm is reasonable and often done in the literature



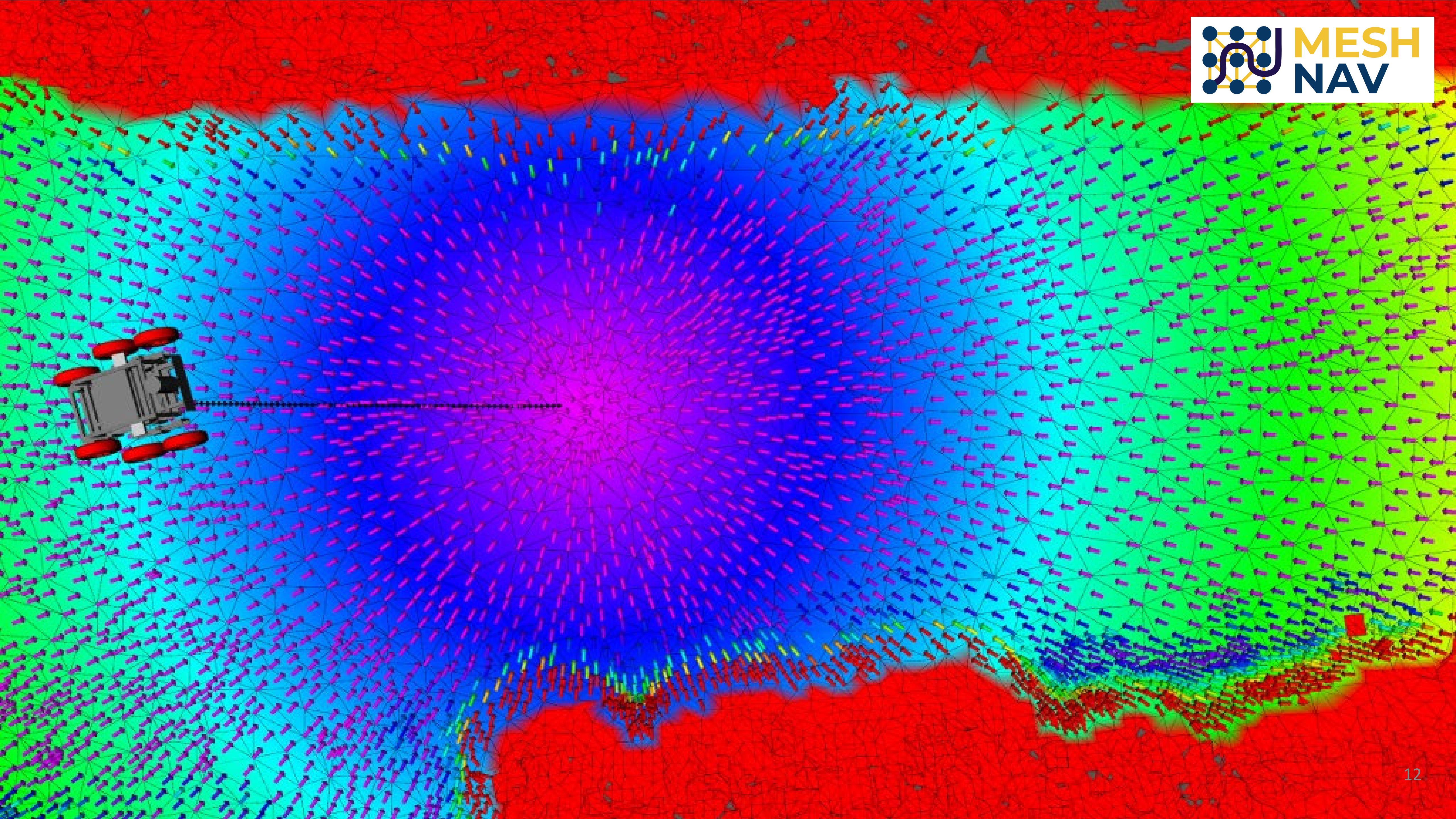
"If p is a geodesic path which connects the edge sequence E then the planar unfolding of p along the edge sequence E is a straight line segment" (1987, Mitchell et al. [199], Lemma 3.3)

Continuous Vector Field Planner

Update Step – Distance and Direction Computation



Sebastian Pütz, Thomas Wiemann, Malte kleine Piening and Joachim Hertzberg, Continuous Shortest Path Vector Field Navigation on 3D Triangular Meshes for Mobile Robots, in 2021 IEEE International Conference on Robotics and Automation (ICRA), May., Xi'an China, 2021



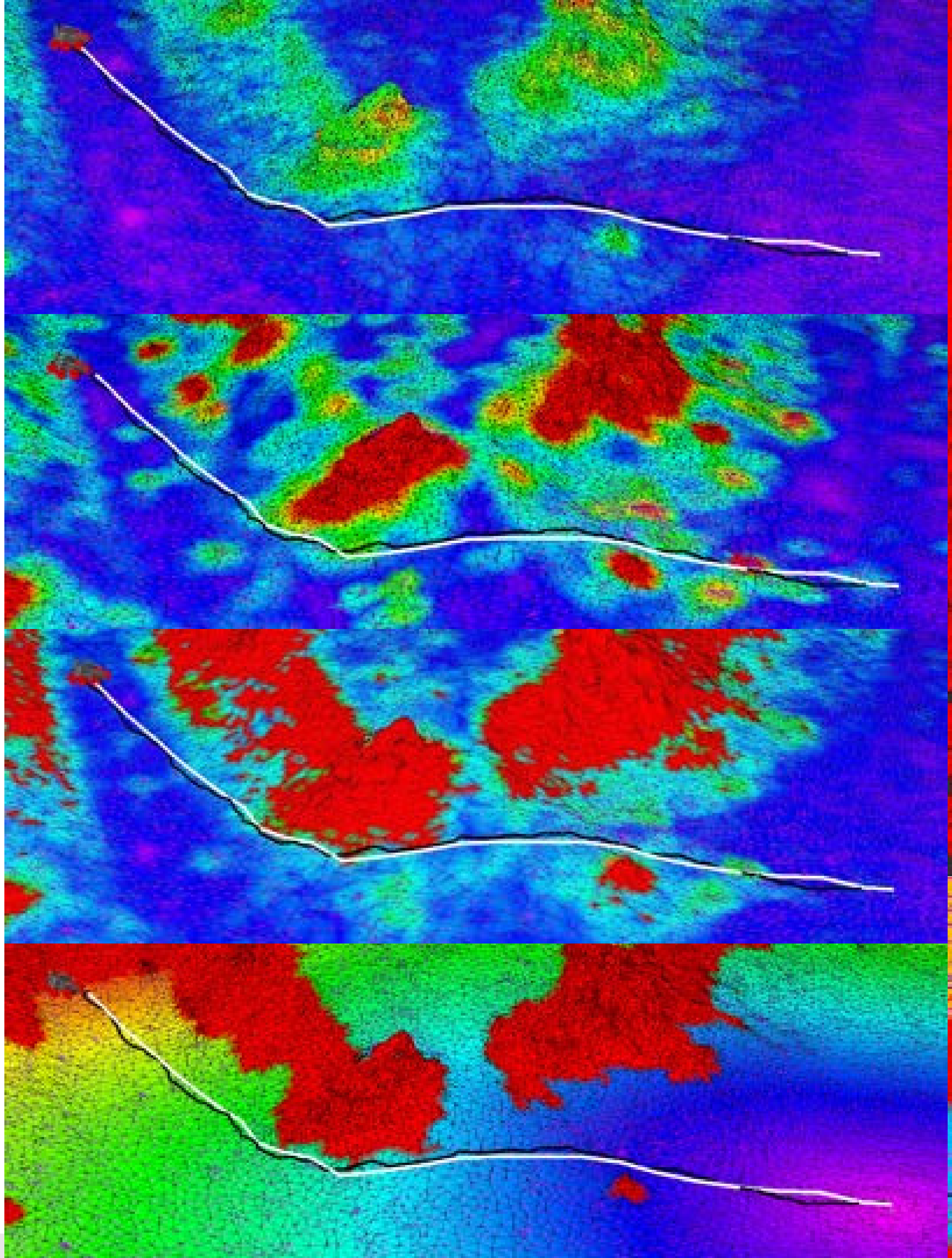
MESH NAV

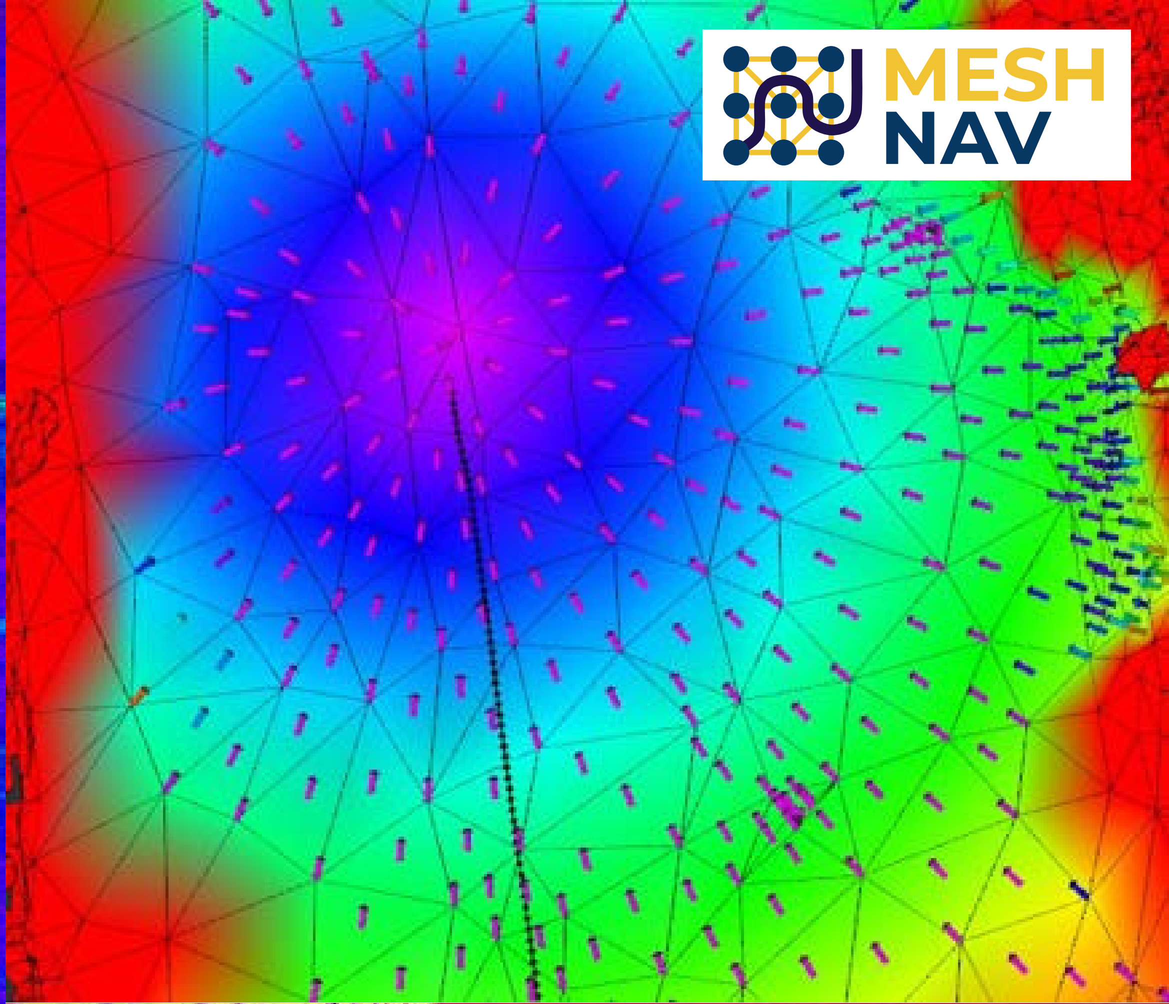
Height Diff

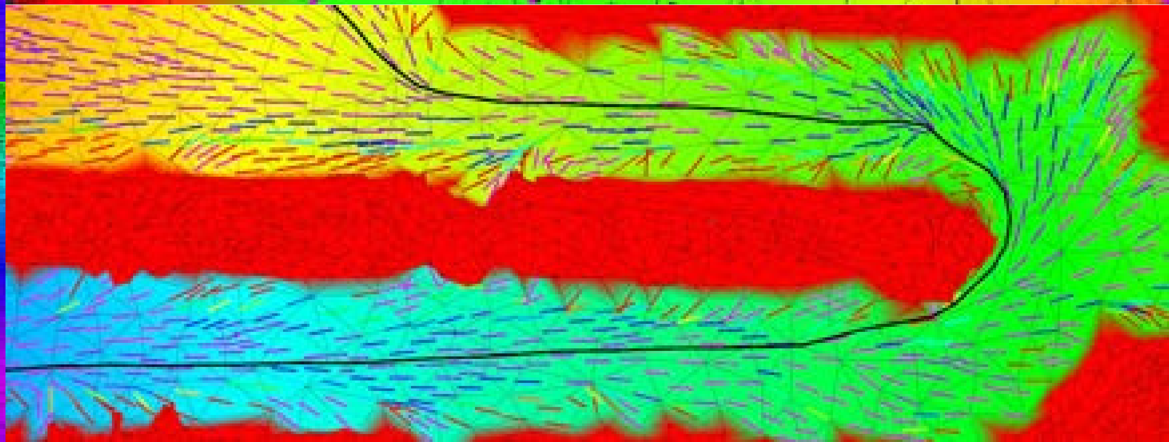
Roughness

Combined

Distance



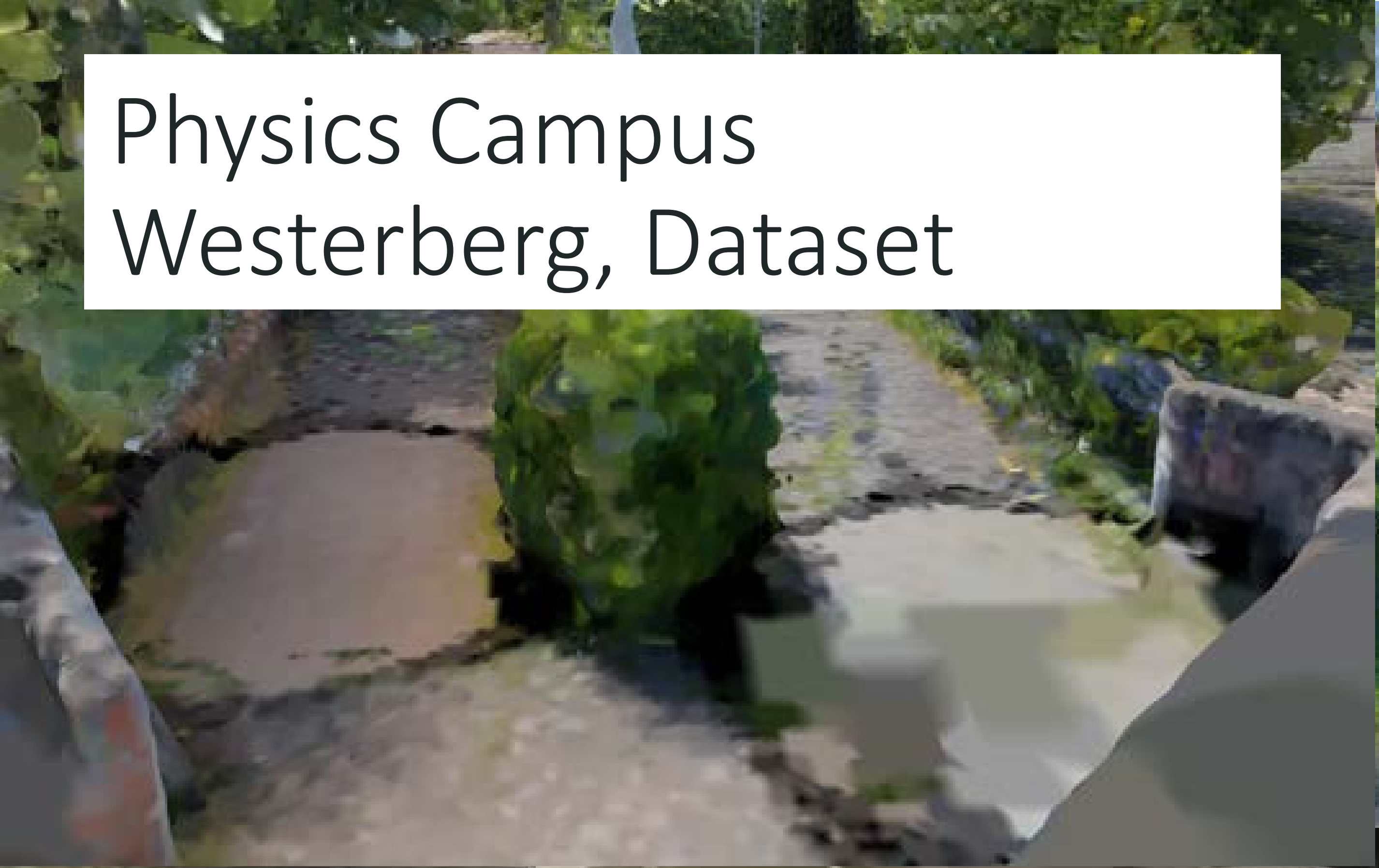




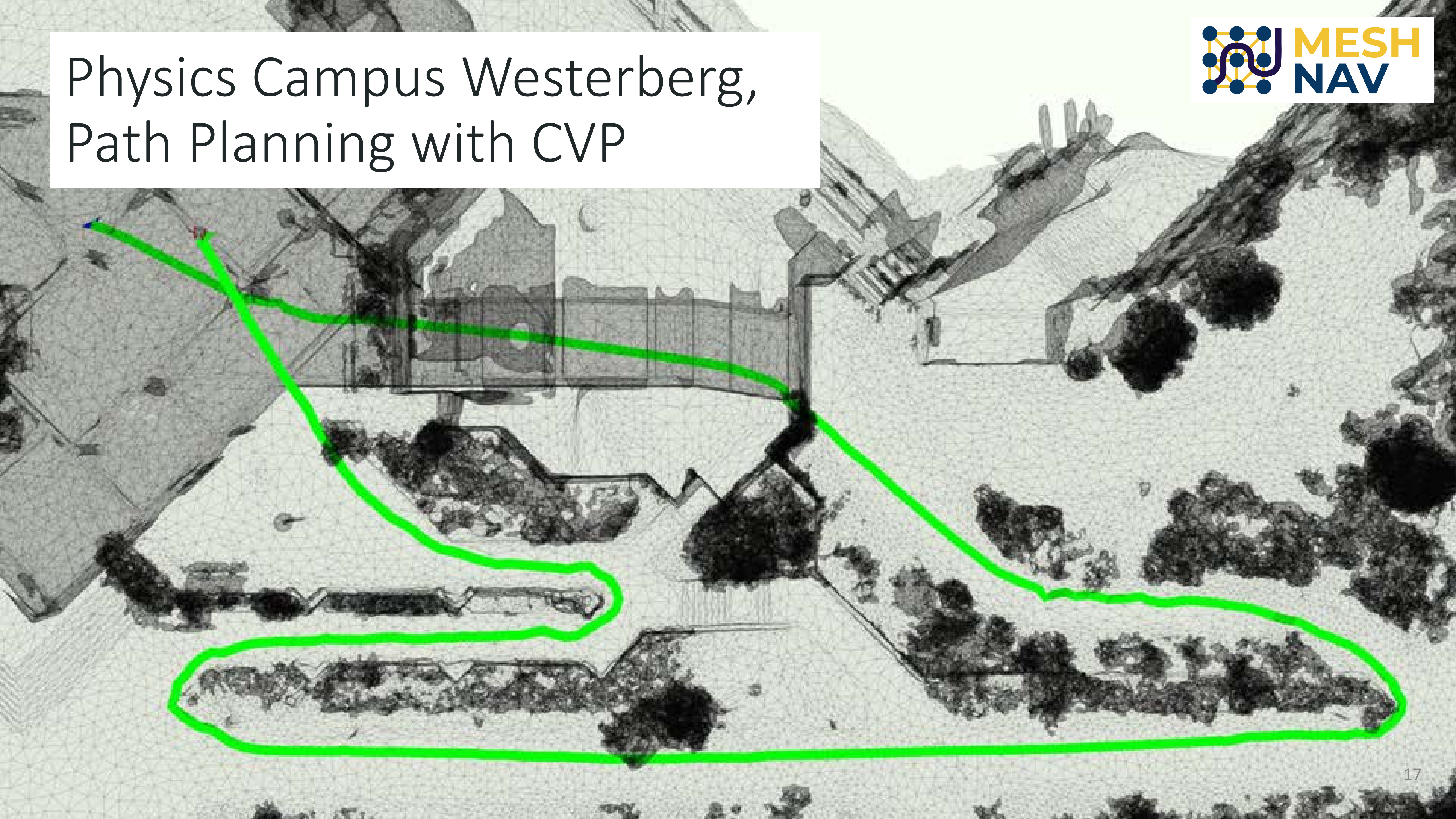
Physics Campus Westerberg, Osnabrück, Germany

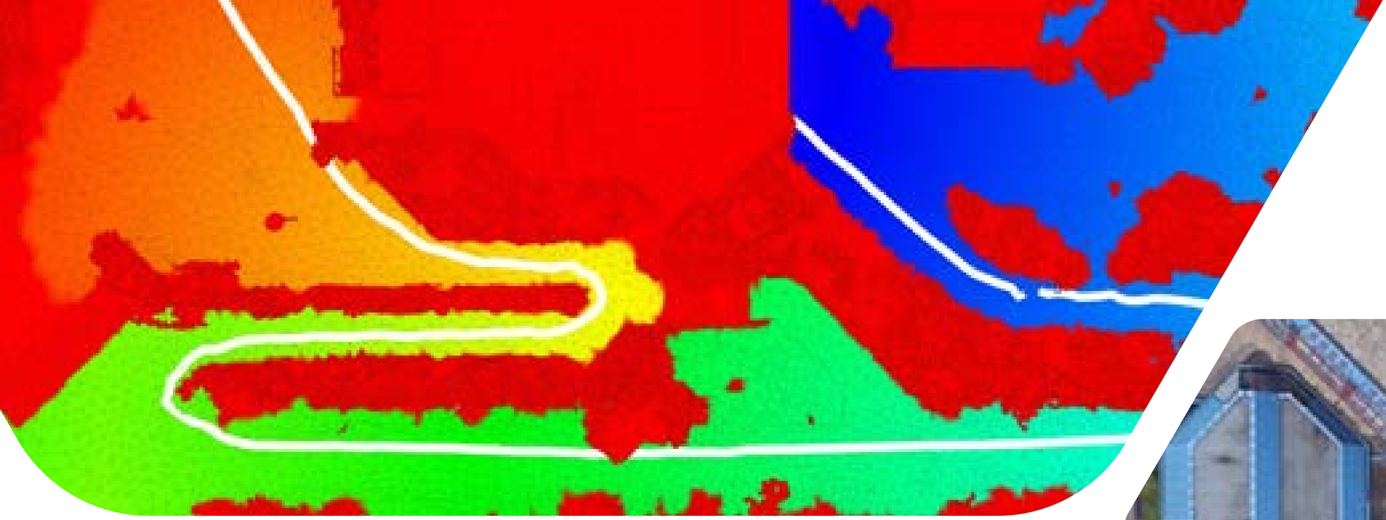
Physics Campus Westerberg, Colored Mesh Dataset

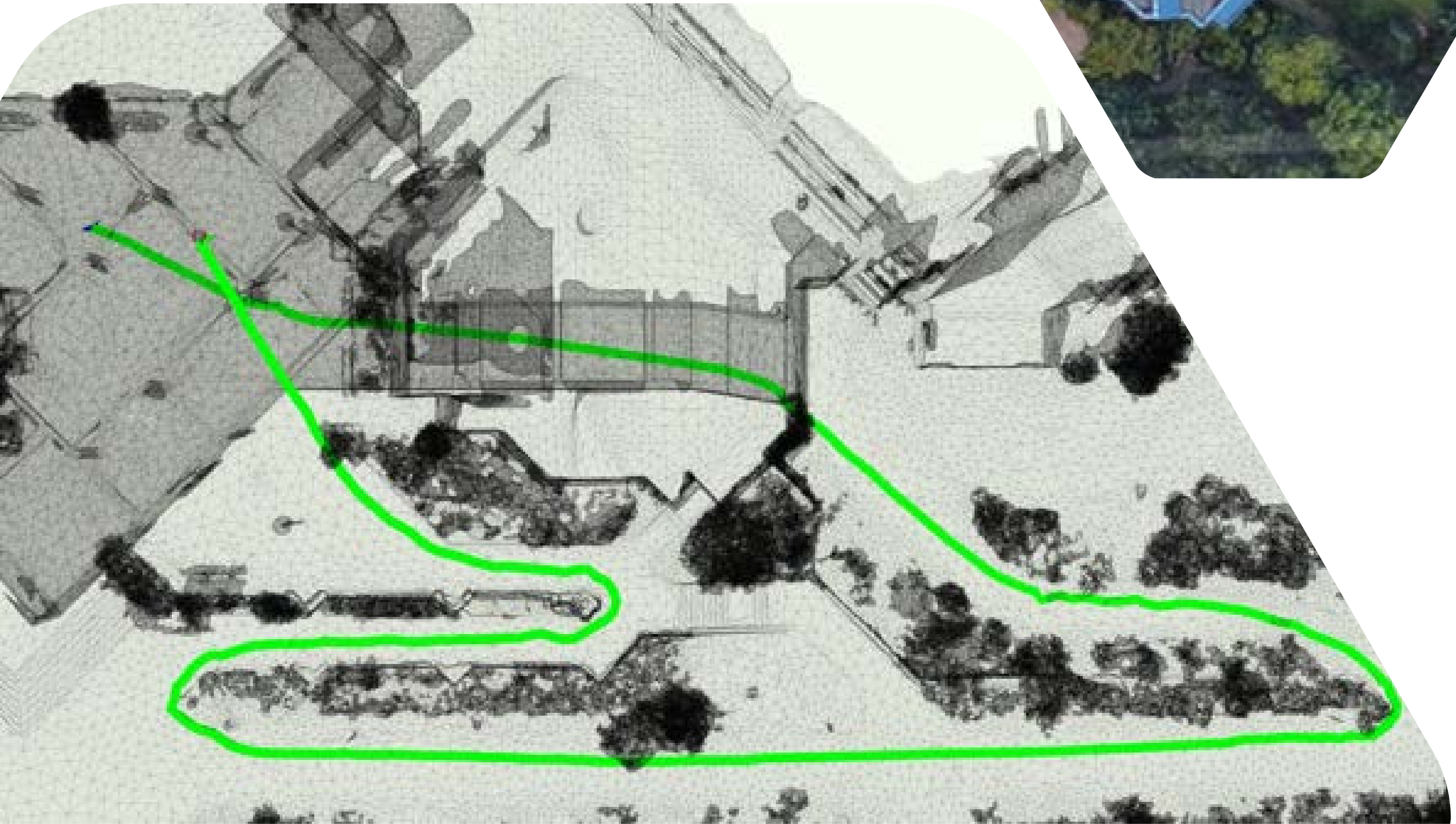
Physics Campus Westerberg, Dataset

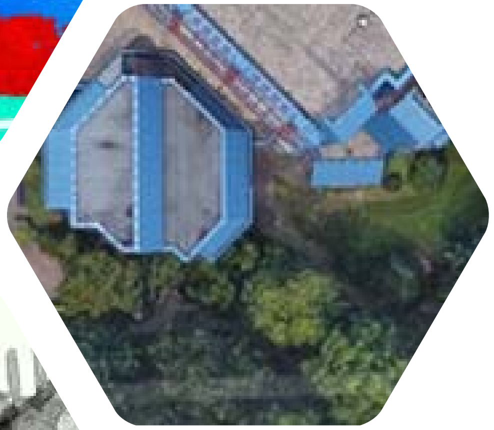


Physics Campus Westerberg, Path Planning with CVP









- multilevel navigation
- through a tunnel and over multiple ramps

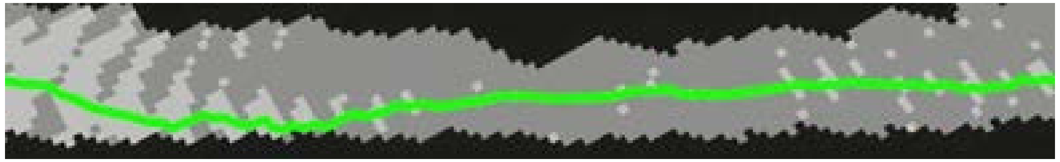
- CVP Path Length Error: 0.77%, 1.54m
- Dijk. Path Length Error: 5.59%, 11.12m

Path Planner	Runtime [ms]	Length [m]
CVP	342.9	200.57
Dijkstra	176.2	210.15
MMP	1 839.5	199.03

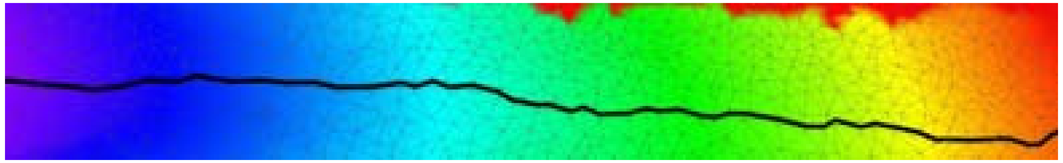
Experimental Results

- 1.8 times more time than Dijkstra, in most cases in under a second
- MMP takes around 21,4 times more time than Dijkstra
- Our CVP Planner produces negligibly longer paths than the exact shortest paths
- Moreover, it computes a continuous vector field towards the goal

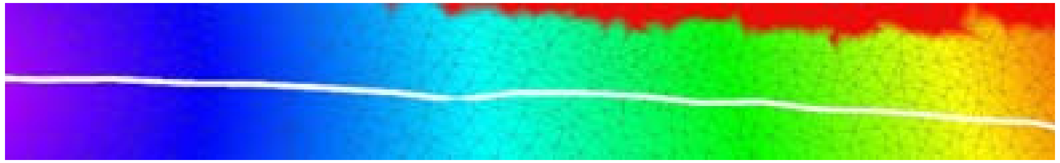
A* / Dijkstra costmap_2d



Dijkstra Mesh Map



CVP Mesh Map



Dataset	# Vertices $\lvert V \rvert$	# Vertices $\lvert V' \rvert$	# Triangles $\lvert F \rvert$	Bounding Box [m] x	Bounding Box [m] y	Bounding Box [m] z	Runtime [ms] CVP	Runtime [ms] Dijk.	Runtime [ms] MMP	Path Length [m] CVP	Path Length [m] Dijk.	Path Length [m] MMP
Botanical Garden Osnabrück	714 760	7 375	1 430 188	39.05	49.25	6.67	108.1	51.4	1 731.8	26.34	27.47	26.09
Stone Quarry Brockum	992 879	208 740	1 904 178	100.58	100.58	23.94	831.0	451.8	7 364.6	110.26	114.67	109.54
Physics Campus Westerberg	718 383	36 776	1 617 772	166.02	83.61	26.33	342.9	176.2	1 839.5	200.57	210.15	199.03
Farmer's Pit Stemwede	401 036	40 623	794 509	122.23	104.57	14.84	56.2	36.1	1 348.3	54.04	56.02	52.93
Market Garden Ibbenbüren	1 361 308	132 210	2 656 283	174.33	149.61	24.58	1 211.0	695.4	6 316.0	95.71	98.62	94.46

Summary

MESH NAV

- Fully integrated with Move Base Flex:
  - Planner, Controller, Recovery Interfaces: mbf_mesh_core
  - MBF Mesh Navigation Server: mbf_mesh_nav
- Layered Mesh Map representation:
  - Trafficability / Drivability Layers can be loaded as plugins
  - Available plugins: Roughness, Height Differences, Steepness, Inflation,
- Mesh Path planners loadable as plugins:
  - CVP, Dijkstra, MMP

- 5 test datasets of real outdoor environments:
  - HDF5 Mesh Maps with layer information
  - Corresponding Gazebo simulations Worlds
- Launch and configuration files to reproduce experimental results
- 360 stars, 50 forks on GitHub
- Continuous Integration
- released to the official ROS repositories: (apt install ros-noetic-mesh-navigation)

RMCL
MESH LOCALIZATION

Localization in Meshes using Range Sensors

Use available range sensor data to localize the robot relative to a given mesh

RMCL
MESH LOCALIZATION

github.com/uos/rmcl, Open-Source!

Computing Units
- CPU
- GPU

Sensors
- LiDAR
- Depth Cam

Environments
- Large-scale
- Outdoors / Indoors

RMCL – Pose Tracking: MICP-L

Given a coarse odometry prior, continuously register range sensor data to the mesh map to track the robot's pose: MICP-L [1]

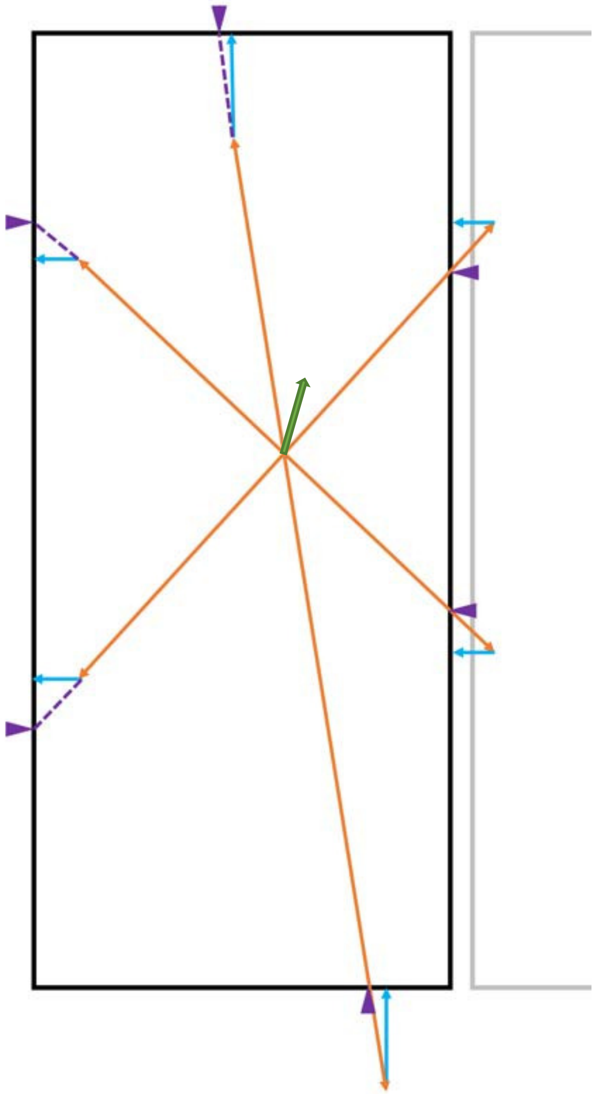
[1] A. Mock, S. Pütz, T. Wiemann, and J. Hertzberg, "MICP-L: Mesh ICP for Robot Localization using Hardware-Accelerated Ray Casting," 2023. arXiv.

Hardware-accelerated Ray Casting Correspondences

Find RCCs:

1. Given a pose estimate: (green arrow)
2. For each scan direction (orange / purple dotted line)
3. Ray cast -> intersection with the map (purple triangle)
4. Correspondence: intersection + normal & scan point (tips of orange arrows)
5. Optimization: point to plane (blue arrow)

Robust if there are walls!



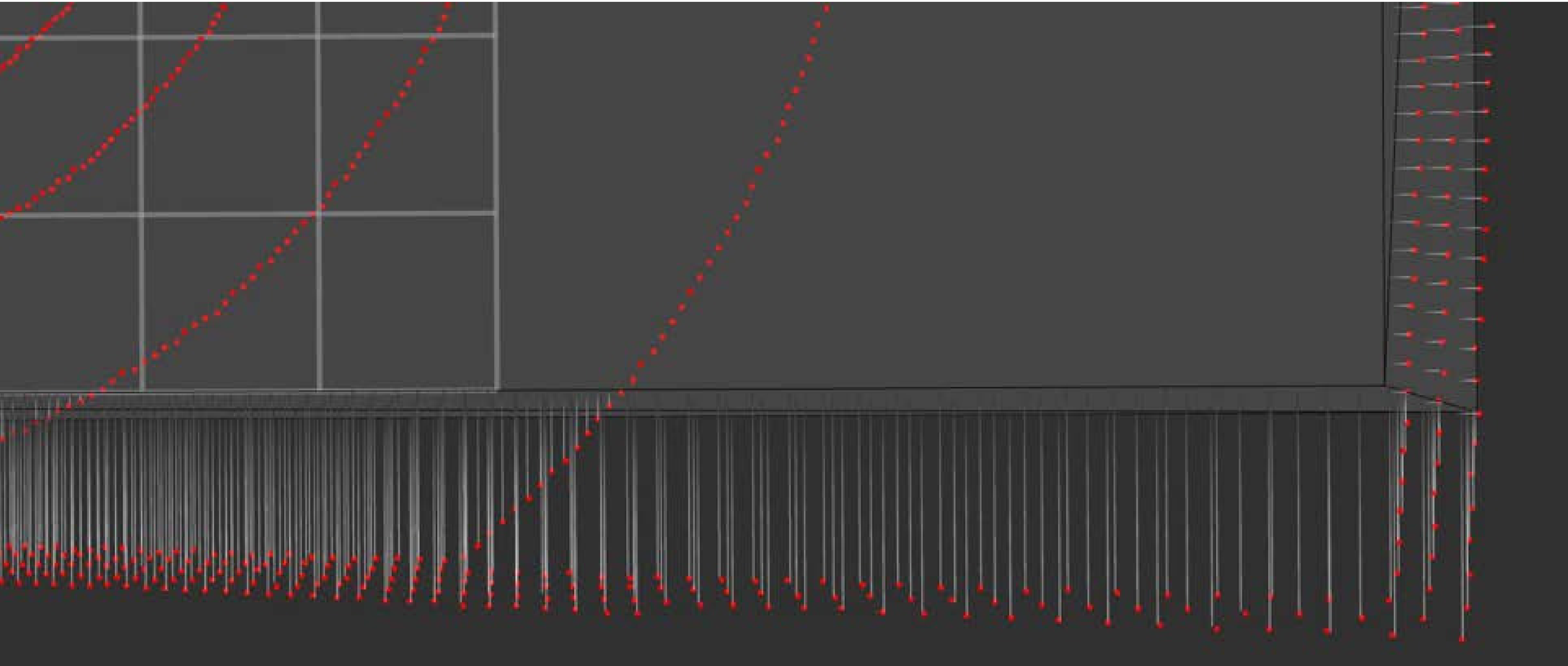
RCCs [2] implemented via Rmagine [3]

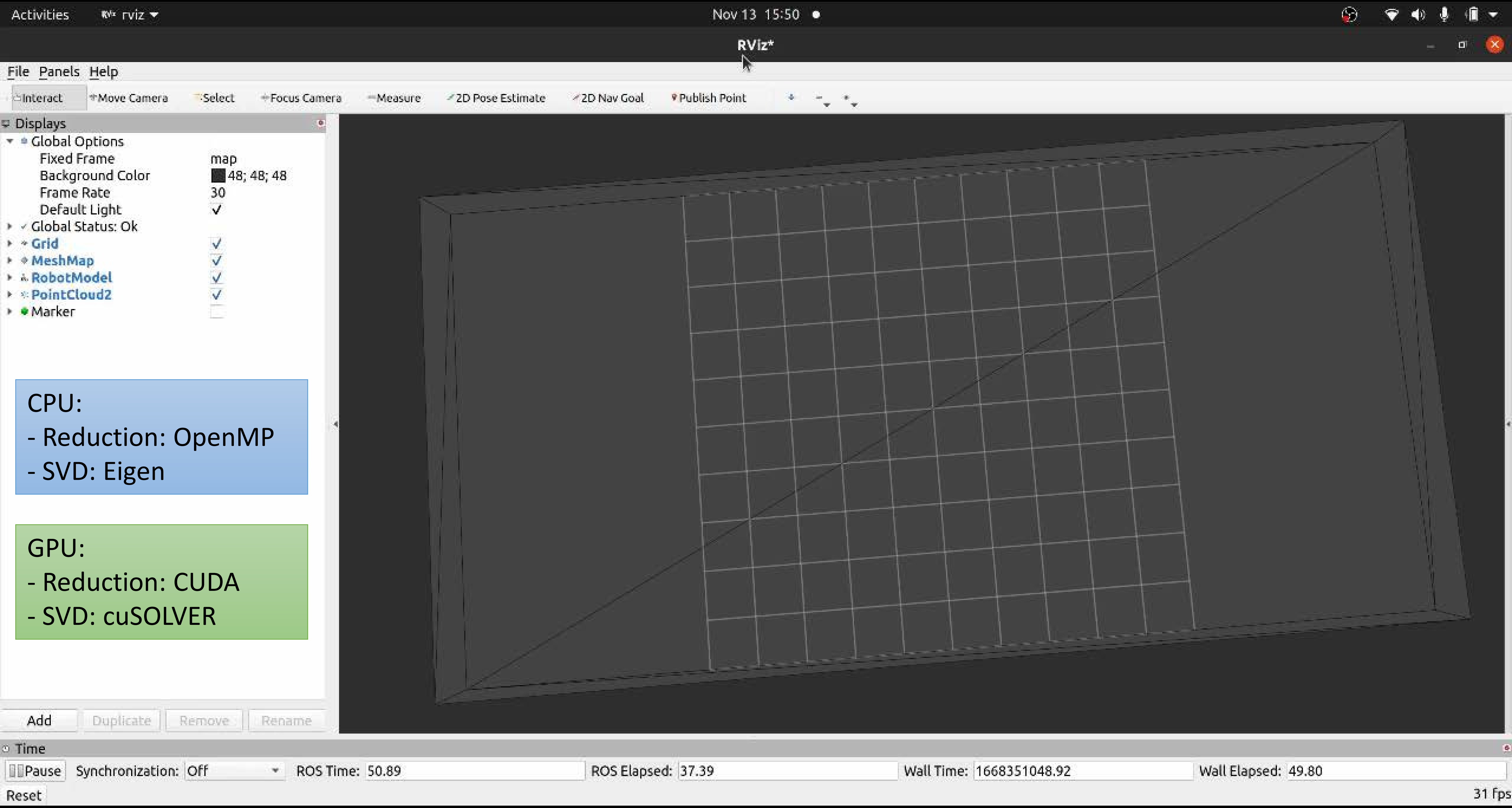
- CPU: Intel Embree
- GPU: NVIDIA OptiX

[2] I. Vizzo, X. Chen, N. Chebrolu, J. Behley, and C. Stachniss, “Poisson Surface Reconstruction for LiDAR Odometry and Mapping,” in International Conference on Robotics and Automation (ICRA). IEEE, 2021, pp. 5624–5630.

[3] A. Mock, T. Wiemann, and J. Hertzberg, “Rmagine: 3D Range Sensor Simulation in Polygonal Maps via Ray Tracing for Embedded Hardware on Mobile Robots,” in International Conference on Robotics and Automation (ICRA). IEEE, 2023, pp. 9076–9082.

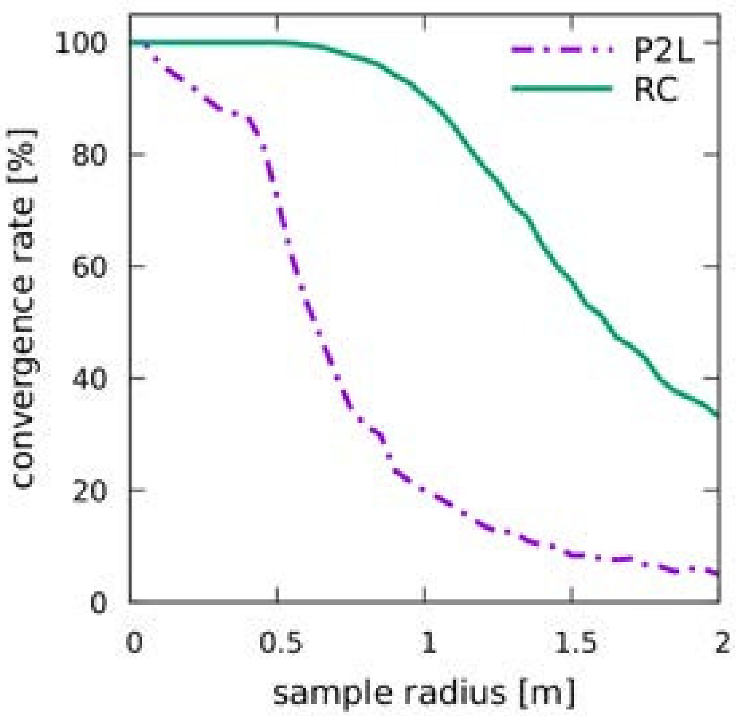
Hardware-accelerated Ray Casting Correspondences

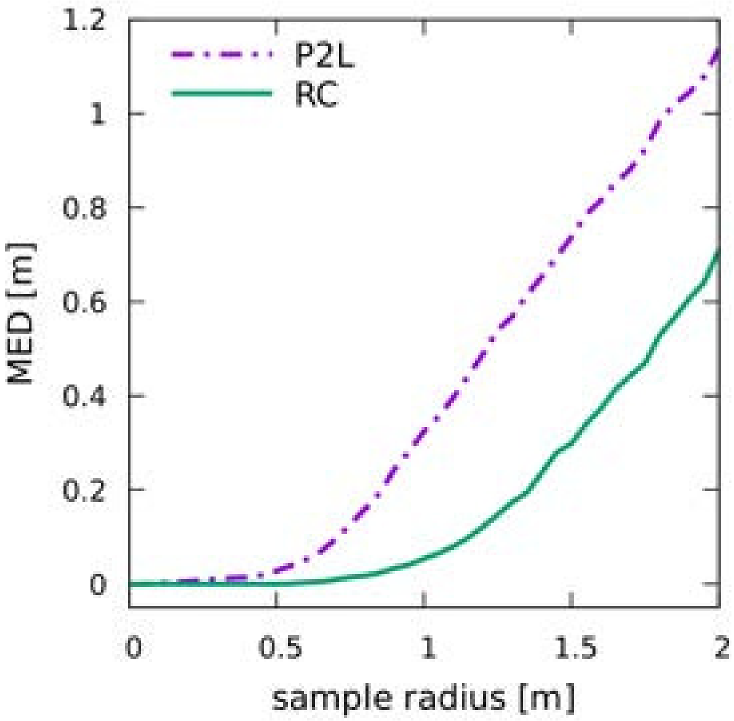



Activities
rviz
Nov 13 15:50
RViz*
File
Panels
Help
Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
2D Nav Goal
Publish Point
Displays
Global Options
Fixed Frame
map
Background Color
48; 48; 48
Frame Rate
30
Default Light
Global Status: Ok
Grid
MeshMap
RobotModel
PointCloud2
Marker
CPU:
- Reduction: OpenMP
- SVD: Eigen
GPU:
- Reduction: CUDA
- SVD: cuSOLVER
Add
Duplicate
Remove
Rename
Time
Pause
Synchronization:
Off
ROS Time:
50.89
ROS Elapsed:
37.39
Wall Time:
1668351048.92
Wall Elapsed:
49.80
Reset
31 fps

Convergence

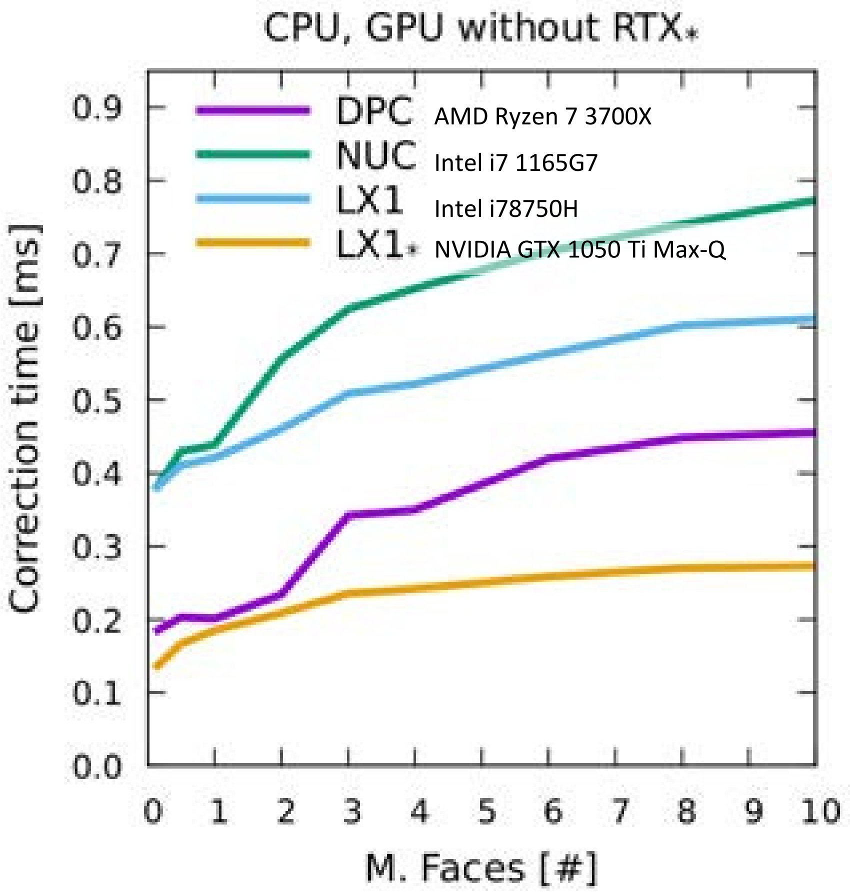
- Ray casting (RC) vs point 2 plane (P2L)
- Sample initial pose guesses
- Measure convergence rate and registration accuracy (MED)

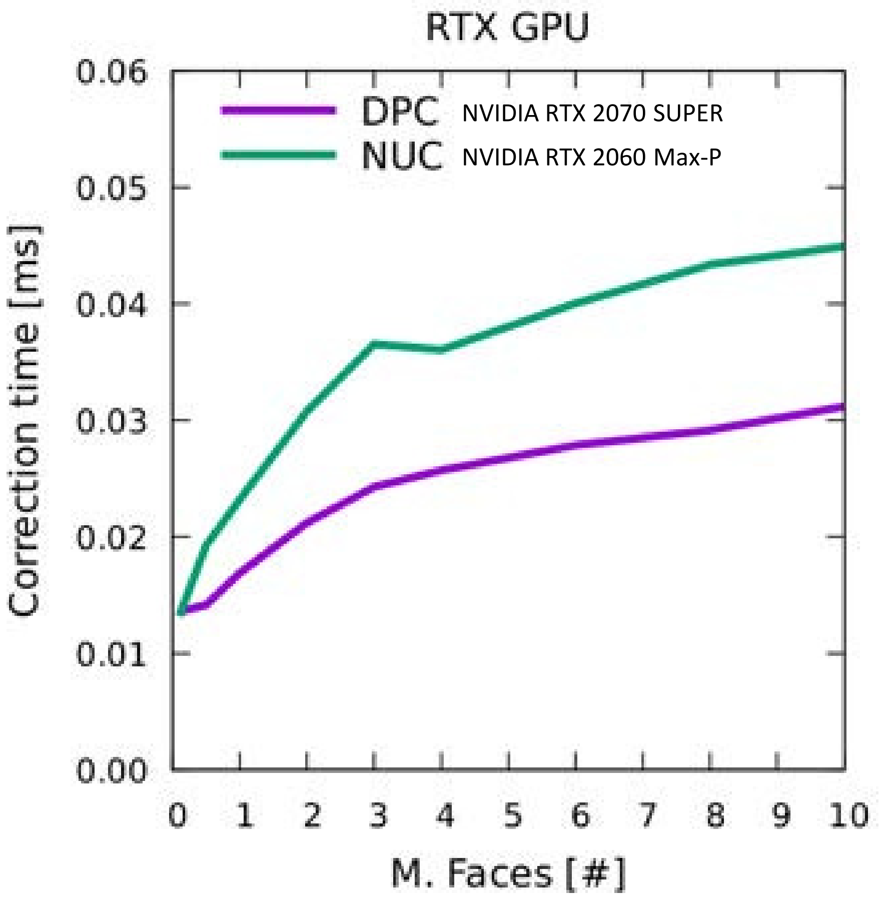




Run time

- VLP-16 – 900 azimuths. 14,400 measurements / scan
- Correct 1000 pose guesses in parallel
  - Plot run time per pose w.r.t. number of faces (sphere)
  - Repeat for various computing devices





Usage

- Example launch and configuration files
- File path to mesh map
- Odometry estimation as tf frame
- Configure sensor models
- Multiple sensors in 2D and 3D
- GPU or CPU

Example cofig file: micp_velodyne_cpu.yaml

```yaml
# required
base_frame: base_footprint
map_frame: map
odom_frame: odom

# rate of broadcasting tf transformations
tf_rate: 50
invert_tf: False

micp:
  # merging on gpu or cpu
  combining_unit: cpu
  # maximum number of correction steps per second
  # lower this to decrease the correction speed but save energy
  corr_rate_max: 1000
  print_corr_rate: False

  # adjust max distance dependend of the state of localization
  # helps to continuously disregard objects that not exist in the map
  adaptive_max_dist: True # enable adaptive max dist

  # offset added to inital pose guess
  trans: [0.0, 0.0, 0.0]
  rot: [0.0, 0.0, 0.0] # euler angles (3) or quaternion (4)
```

```yaml
# describe your sensor setup here
sensors: # list of range sensors - at least one is
  velodyne:
    topic: velodyne_points
    # spherical is comparable to sensor_msgs::Laser
    # but in 3D
    type: spherical
    model:
      range_min: 0.5
      range_max: 130.0
      phi_min: -0.261799067259
      phi_inc: 0.03490658503988659
      phi_N: 16
      theta_min: -3.14159011841
      theta_inc: 0.01431249500496489
      theta_N: 440
    micp:
      max_dist: 1.0
      # Once adaptive_max_dist is set to true:
      #
      # If the localization is perfect, the max
      # distance for finding SPCs is reduced to
      # `adaptive_max_dist_min`.
      # If the localization is bad, the max
      # distance for finding SPCs is raised to
      # `max_dist`
      adaptive_max_dist_min: 0.15
      backend: embree
```

Example launch file: my_micp_localization.launch

```xml
<launch>

<arg name="map" default="$(find uos_gazebo_worlds)/Media/models/avz_neu.dae" />
<arg name="config" default="$(find rmcl)/config/examples/micp_velodyne_cpu.yaml" />

<node pkg="rmcl" type="micp_localization" name="micp_localization" output="screen">
    <param name="map_file" type="string" value="$(arg map)" />
    <rosparam command="load" file="$(arg config)" />
    <remap from="pose_wc" to="/initialpose" />
</node>

</launch>
```

Mesh Navigation & Mesh Localization

MESH NAV

RMCL MESH LOCALIZATION

Code

github.com/naturerobots/mesh_navigation

Videos

youtube.com/@nature-robots

Code

github.com/uos/rmcl

Videos

youtube.com/@uos-kbs

References – Mesh Navigation

[35] Marcel Campen, Martin Heistermann, and Leif Kobbelt. Practical Anisotropic Geodesy. Computer Graphics Forum, 32(5):63–71, August 2013. ISSN 01677055. doi: 10.1111/cgf.12173.

[38] Jindong Chen and Yijie Han. Shortest Paths on a Polyhedron. In Proceedings of the Sixth Annual Symposium on Computational Geometry, SCG '90, pages 360–369, New York, NY, USA, May 1990. Association for Computing Machinery. ISBN 978-0-89791-362-1. doi: 10.1145/98524.98601.

[51] Keenan Crane, Clarisse Weischedel, and Max Wardetzky. Geodesics in Heat: A New Approach to Computing Distance Based on Heat Flow. ACM Transactions on Graphics, 32(5):1–11, September 2013. ISSN 07300301. doi: 10.1145/2516971.2516977.

[90] Zhisong Fu, Won-Ki Jeong, Yongsheng Pan, Robert M. Kirby, and Ross T. Whitaker. A Fast Iterative Method for Solving the Eikonal Equation on Triangulated Surfaces. SIAM Journal on Scientific Computing, 33(5):2468–2488, 2011. doi: 10.1137/100788951.

[136] Takashi Kanai and Hiromasa Suzuki. Approximate Shortest Path on a Polyhedral Surface Based on Selective Refinement of the Discrete Graph and Its Applications. In Geometric Modeling and Processing 2000. Theory and Applications. Proceedings, pages 241–250, Hong Kong, China, 2000. IEEE. ISBN 978-0-7695-0562-6. doi: 10.1109/GMAP.2000.838256.

[199] (Slide 17) Joseph S. B. Mitchell, David M. Mount, and Christos H. Papadimitriou. The Discrete Geodesic Problem. SIAM Journal on Computing, 16(4):647–668, August 1987. doi: 10.1137/0216045.

[119] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. Rgb-D Mapping: Using Kinect-Style Depth Cameras for Dense 3d Modeling of Indoor Environments. International Journal of Robotic Research - IJRR, 31:647–663, April 2012. doi: 10.1177/0278364911434148.

[152] Ron Kimmel and James A Sethian. Computing Geodesic Paths on Manifolds. Proceedings of the National Academy of Sciences, 95(15):8431–8435, 1998. doi: 10.1073/pnas.95.15. 8431.

[153] Danil Kirsanov and Steven J Gortler. Minimal Discrete Curves and Surfaces. PhD Thesis, Harvard University Cambridge, MA, 2004.

[154] Danil Kirsanov, Steven J. Gortler, and Hugues Hoppe. Fast Exact and Approximate Geodesic Paths on Meshes. Harvard University Computer Science TR 10, 4, 2004.

[171] Mark Lanthier. Shortest Path Problems on Polyhedral Surfaces. PhD Thesis, School of Computer Science, Carleton University, Ottawa, Ontario, December 1999.

[172] Mark Lanthier, Anil Maheshwari, and Jörg-Rüdiger Sack. Approximating Weighted Shortest Paths on Polyhedral Surfaces. In Proceedings of the Thirteenth Annual Symposium on Computational Geometry, pages 274–283, 1997.

[199] Joseph S. B. Mitchell, David M. Mount, and Christos H. Papadimitriou. The Discrete Geodesic Problem. SIAM Journal on Computing, 16(4):647–668, August 1987. ISSN 0097-5397, 1095-7111. doi: 10.1137/0216045.

[210] Marcin Novotni and Reinhard Klein. Computing Geodesic Distances on Triangular Meshes. Journal of WSCG, 10(1-2):341–347, 2002.

[218] Joseph O'Rourke, Subhash Suri, and Heather Booth. Shortest Paths on Polyhedral Surfaces. In K. Mehlhorn, editor, STACS 85, pages 243–254, Berlin, Heidelberg, 1985. Springer Berlin Heidelberg. ISBN 978-3-540-39136-4.

[236] Jianliang Qian, Yong-Tao Zhang, and Hong-Kai Zhao. Fast Sweeping Methods for Eikonal Equations on Triangular Meshes. SIAM Journal on Numerical Analysis, 45(1):83–107, January 2007. ISSN 0036-1429, 1095-7170. doi: 10.1137/050627083.

[276] Micha Sharir and Amir Schorr. On Shortest Paths in Polyhedral Spaces. SIAM Journal on Computing, 15(1):193–215, 1986. doi: 10.1137/0215014.

[290] Vitaly Surazhsky, Tatiana Surazhsky, Danil Kirsanov, Steven J. Gortler, and Hugues Hoppe. Fast Exact and Approximate Geodesics on Meshes. ACM Transactions on Graphics (TOG), 24(3):553–560, 2005. doi: 10.1145/1073204.1073228.

[313] Shi-Qing Xin and Guo-Jin Wang. Improving Chen and Han's Algorithm on the Discrete Geodesic Problem. ACM Transactions on Graphics (TOG), 28(4):1–8, 2009.

[316] Chunxu Xu, Tuanfeng Y. Wang, Yong-Jin Liu, Ligang Liu, and Ying He. Fast Wavefront Propagation (FWP) for Computing Exact Geodesic Distances on Meshes. IEEE Transactions on Visualization and Computer Graphics, 21(7):822–834, July 2015. ISSN 1941-0506. doi: 10.1109/TVCG.2015.2407404.

[320] Xiang Ying, Xiaoning Wang, and Ying He. Saddle Vertex Graph (SVG): A Novel Solution to the Discrete Geodesic Problem. ACM Transactions on Graphics (TOG), 32, November 2013. doi: 10.1145/2508363.2508379.

[321] Xiang Ying, Shi-Qing Xin, and Ying He. Parallel Chen-Han (PCH) Algorithm for Discrete Geodesics. ACM Transactions on Graphics (TOG), 33(1):1–11, 2014. doi: 10.1145/2534161.

References – Mesh Localization

[1] A. Mock, S. Pütz, T. Wiemann, and J. Hertzberg, "MICP-L: Mesh ICP for Robot Localization using Hardware-Accelerated Ray Casting," 2023. arXiv.

[2] I. Vizzo, X. Chen, N. Chebrolu, J. Behley, and C. Stachniss, "Poisson Surface Reconstruction for LiDAR Odometry and Mapping," in International Conference on Robotics and Automation (ICRA). IEEE, 2021, pp. 5624–5630.

[3] A. Mock, T. Wiemann, and J. Hertzberg, "Rmagine: 3D Range Sensor Simulation in Polygonal Maps via Ray Tracing for Embedded Hardware on Mobile Robots," in International Conference on Robotics and Automation (ICRA). IEEE, 2023, pp. 9076–9082.

[4] I. Wald, S. Woop, C. Benthin, G. S. Johnson, and M. Ernst, "Embree: A Kernel Framework for Efficient CPU Ray Tracing," ACM Transactions on Graphics (TOG), 2014.

[5] S. G. Parker, J. Bigler, A. Dietrich, H. Friedrich, J. Hoberock, D. Luebke, D. McAllister, M. McGuire, K. Morley, A. Robison, and M. Stich, "OptiX: A General Purpose Ray Tracing Engine," ACM Transactions on Graphics (TOG), 2010.

[6] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," IEEE Pattern Analysis & Machine Intelligence (PAMI), vol. 13, no. 04, pp. 376–380, 1991.

[7] M. Helmberger, K. Morin, B. Berner, N. Kumar, G. Cioffi, and D. Scaramuzza, "The Hilti SLAM Challenge Dataset," IEEE Robotics and Automation Letters (RAL), vol. 7, no. 3, pp. 7518–7525, 2022.

[8] G. Kim, Y. S. Park, Y. Cho, J. Jeong, and A. Kim, "MulRan: Multimodal Range Dataset for Urban Place Recognition," in International Conference on Robotics and Automation (ICRA). IEEE, 2020, pp. 6246–6253.