Structural Inspection Coverage Path Planner

CS686, Paper Presentation

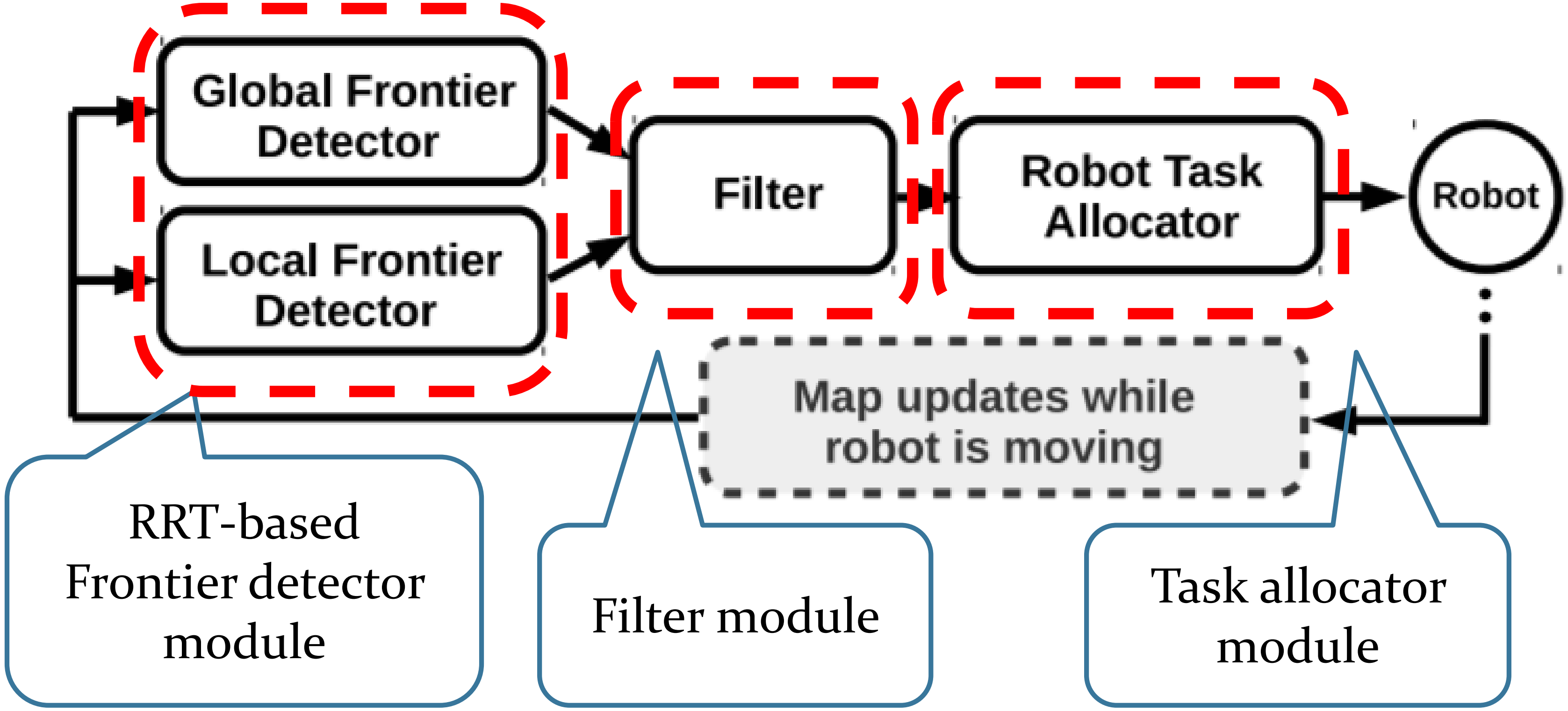
유병호(Yu Byeong-ho)

Andreads, B. et al., "Structural Inspection Path Planning via Iterative Viewpoint Resampling with Application to Aerial Robotics," ICRA 2015.

Sungwook, J. et al., "Multi-layer Coverage Path Planner for Autonomous Structural Inspection of High-rise Structures," IROS 2018.

Recap.

Autonomous Robotic Exploration Based on Multiple RRT



"Detecting frontier points"

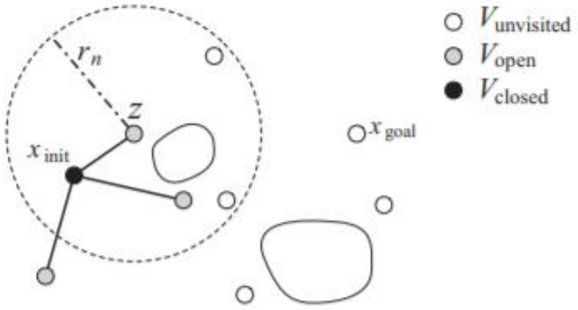
"Clustering the frontier points"

"Assigning a target point"

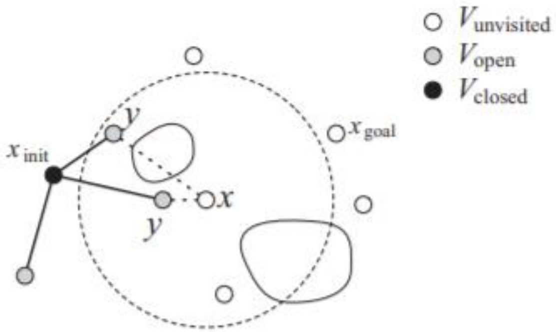
- Overall schematic diagram of the exploration algorithm -

Reference: Hassan Umari, Autonomous robotic exploration based on multiple rapidly-exploring randomized trees (IROS 2017)

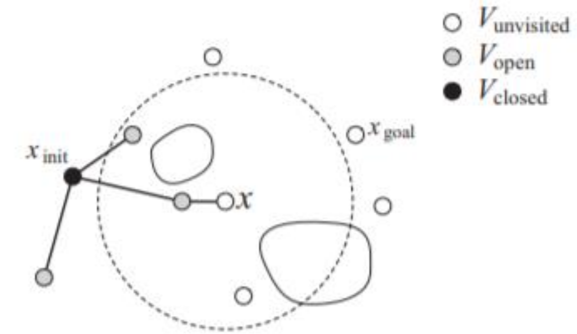
Fast Marching Tree



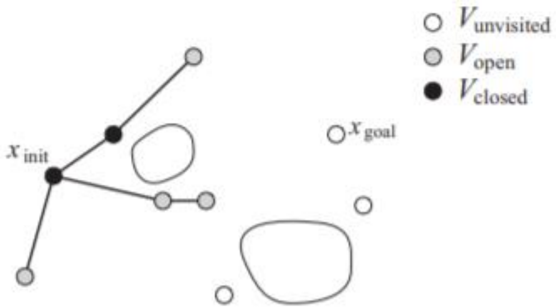
(a) Lines 2–3: FMT* selects the lowest-cost node z from set V_{open} and finds its neighbors within $V_{unvisited}$.



(b) Lines 4–5: given a neighboring node x, FMT* finds the neighbors of x within V_{open} and searches for a locally optimal one-step connection. Note that paths intersecting obstacles are also lazily considered.



(c) Line6:FMT* selects the locally optimal one-step connection to x ignoring obstacles, and adds that connection to the tree if it is collision-free.



(d) Lines 7–8: After all neighbors of z in $V_{unvisited}$ have been explored, FMT* adds successfully connected nodes to V_{open}, places z in V_{closed}, and moves to the next iteration.

Reference: Lukas Janson, Fast Marching Tree: A fast marching sampling-based method for optimal motion planning in many dimensions (IJRR 2015)

1.

Structural Inspection Path Planning via Iterative Viewpoint Resampling with Application to Aerial Robotics


Structural Inspection Path Planning via Iterative Viewpoint Resampling with Application to Aerial Robotics
Andreas Bircher, Kostas Alexis, Michael Burri, Phlipp Oettershagen, Sammy Omari, Thomas Mantel and Roland Siegwart
ETH

Introduction

- In the fields of inspection operations, autonomous complete coverage 3D structural path planning is required.

- A robot needs fast algorithms that result in full coverage of the structure while respecting any sensor limitations and motion constraints.
  - Especially, drones are limited due to its payload.

- **A novel fast algorithm that provides efficient solutions to the problem of inspection path planning for complex 3D structures is proposed.**

Problem description

- Conventional 3D methods:
  - Two-step optimization
    - Compute the minimal set of viewpoints that cover whole structure (solving Art Gallery Problem (AGP[1])).
    - Compute the shortest connecting tour over these viewpoints (Travelling Salesman Problem (TSP[2])).
  - Large cost of computing efficiency (*expensive*)
  - They are prone to be suboptimal due to the two-step separation of the problem.
  - In specific cases they can lead to unfeasible solutions/paths (e.g. in the case of non-holonomic vehicles)

[1] J. O'rourke, Art gallery theorems and algorithms. Oxford University Press Oxford, 1987, vol. 57.

[2] G. Dantzig, R. Fulkerson, and S. Johnson, "Solution of a large- scale traveling-salesman problem," Journal of the operations research society of America, vol. 2, no. 4, pp. 393–410, 1954.

Contributions

- Not minimizing the number of viewpoints, it samples them such that connecting path is **short** while ensuring **full coverage**

- A two step optimization paradigm to find good viewpoints that together provide full coverage and a connecting path that has low cost
  - First: In every iteration, each viewpoints is chosen to reduce the cost-to-travel between itself and its neighbors
  - Second: the optimally connecting tour is recomputed

Methodology: Algorithm with pseudo-code

Algorithm 1 Inspection path planner

$k \leftarrow 0$
Sample initial viewpoint configurations
Compute cost matrix for the TSP solver (Section IV-A)
Solve the TSP problem to obtain initial tour
while running
    Resample viewpoint configurations (Section IV-B)
    Recompute the cost matrix (Section IV-A)
    Recompute best tour T_{best} using the LKH and update
        best tour cost c_{best} if applicable
    $k \leftarrow k + 1$
end while
return T_{best}, c_{best}

1. Load the mesh model
2. k = 0
3. Sample Initial Viewpoint Configurations (Viewpoint Sampler)
4. Find a Collision-free path for all possible viewpoint combinations (boundary value solver (BVS), RRT*)
5. Compute the Cost Matrix and Solve the Traveling Salesman Problem (Lin-Kernighan-Helsgaun Heuristic (LKH))
6. While running
    1. Re-sample Viewpoint Configurations (Viewpoint Sampler)
    2. Re-compute the Collision-free paths (BVS, RRT*)
    3. Re-populate the Cost Matrix and solve the new Traveling Salesman Problem to update the current best inspection tour (LKH)
    4. k = k + 1
7. end while
8. Return BestTour, CostBestTour

Methodology: Path computation and Cost estimation

- To find the best tour among viewpoints, TSP solver requires a *cost matrix* of all pairs of viewpoints
- Path generation and cost estimation is done by either
  - BVS - directly connect the two viewpoints
  - BVS+RRT* - due to obstacles, connection is not feasible

- The cost of a path segment corresponds to the execution time $\boldsymbol{t_{ex}}$
  - » $\boldsymbol{t_{ex} = \max\left(d/v_{max}\,,\|\psi_1 - \psi_0\|/\dot{\psi}_{max}\right)}$

Where $\boldsymbol{d}$ is the Euclidean distance, translation speed limit is $\boldsymbol{v_{max}}$, rotational speed limit is $\boldsymbol{\dot{\psi}_{max}}$, and $\boldsymbol{\psi}$ is yaw angle, respectively

Methodology: Viewpoint sampling(1)

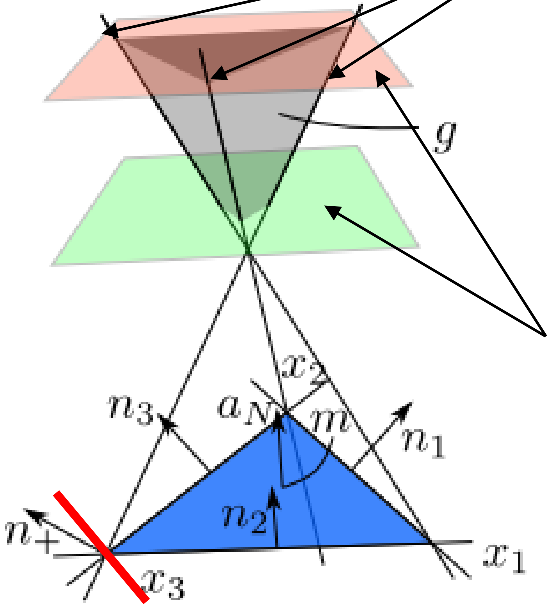
- For every triangle in the mesh, one viewpoint has to be sampled, the position and heading is determined while retaining visibility of the corresponding triangle.

- First, the position is optimized for distance to the neighboring viewpoints using a convex problem formulation and then heading is optimized.

- To guarantee a good result, the position solution must be constrained such as to allow finding an orientation for which the triangle is visible.

Methodology: Viewpoint sampling(2)

- The constraints on the position $\boldsymbol{g} = [\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}]$ consist of the inspection sensor limitation of minimum incidence angle, minimum and maximum range $(\boldsymbol{d_{min}}, \boldsymbol{d_{max}})$ constraints.
  - $$\begin{bmatrix} (g-x_i)^T n_i \\ (g-x_1)^T a_N \\ -(g-x_1)^T a_N \end{bmatrix} \succeq \begin{bmatrix} 0 \\ d_{min} \\ -d_{max} \end{bmatrix}, i = \{1,2,3\}$$
  - Where $\boldsymbol{x_i}$ are the corner of the mesh triangle, $\boldsymbol{a_N}$ is the normalized triangle normal and $\boldsymbol{n_i}$ are the normal of the separating hyperplanes for incidence angle constraints, respectively.



Three main planar angle of incidence constraints on all three sides of the triangle. For a finite number of such constraints the incidence angle is only enforced approximately.
The red line (and $\boldsymbol{n_+}$) demarks a sample orientation for a possible additional planar constraint at a corner. Minimum (green plane) and maximum (red plane) distance constraints are similar planar constraints on the sampling area.
These constraints bound the sampling space, where g can be chosen, on all sides (gray area).

Incidence angle constraints on a triangular facet

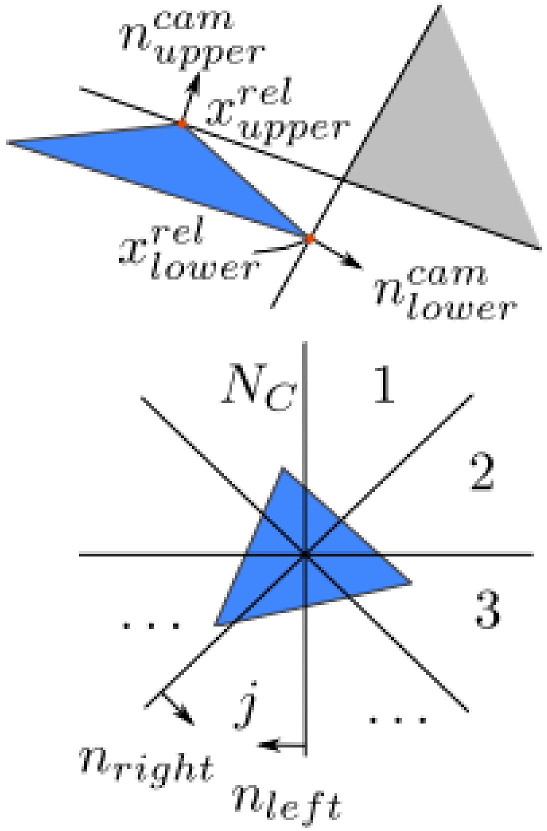
Methodology: Viewpoint sampling(3)

- To account for the limited FoV with fixed pitch angle of camera, it imposes a revoluted 2D-cone constraint which is nonconvex problem and then convexified by dividing the space into $\boldsymbol{N_C}$ equal convex pieces.

- The optimum is computed for every slice.

$$\begin{bmatrix} (g - x^{rel}_{lower})^T n^{cam}_{lower} \\ (g - x^{rel}_{upper})^T n^{cam}_{upper} \\ (g - m)^T n_{right} \\ (g - m)^T n_{left} \end{bmatrix} \succeq \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

  - where $\boldsymbol{x^{rel}_{lower}}$, $\boldsymbol{x^{rel}_{upper}}$ are the respective relevant corners of the mesh triangle, m the middle of the triangle and $\boldsymbol{n^{cam}_{lower}}$, $\boldsymbol{n^{cam}_{upper}}$, $\boldsymbol{n_{right}}$ and $\boldsymbol{n_{left}}$ denote the normal of the respective separating hyperplanes.



Methodology: Viewpoint sampling(4)

- Optimization objective is to minimize the sum of squared distances to the preceding viewpoint $\boldsymbol{g_p^{k-1}}$, the subsequent viewpoint $\boldsymbol{g_s^{k-1}}$ and the current viewpoint in the old tour $\boldsymbol{g^{k-1}}$.

$$\min_{g^k} \quad (g^k - g_p^{k-1})^T (g^k - g_p^{k-1}) + \qquad (3)$$

$$(g^k - g_s^{k-1})^T (g^k - g_s^{k-1}) + (g^k - g^{k-1})^T (g^k - g^{k-1})$$

$$\text{s.t.} \quad \begin{bmatrix} n_1^T \\ n_2^T \\ n_3^T \\ a_N^T \\ -a_N^T \\ n_{lower}^{cam\ T} \\ n_{upper}^{cam\ T} \\ n_{right}^T \\ n_{left}^T \end{bmatrix} g^k \succeq \begin{bmatrix} n_1^T x_1 \\ n_2^T x_2 \\ n_3^T x_3 \\ a_N^T x_1 + d_{min} \\ -a_N^T x_1 - d_{max} \\ n_{lower}^{cam\ T} x_{lower}^{rel} \\ n_{upper}^{cam\ T} x_{upper}^{rel} \\ n_{right}^T m \\ n_{left}^T m \end{bmatrix} \qquad (4)$$

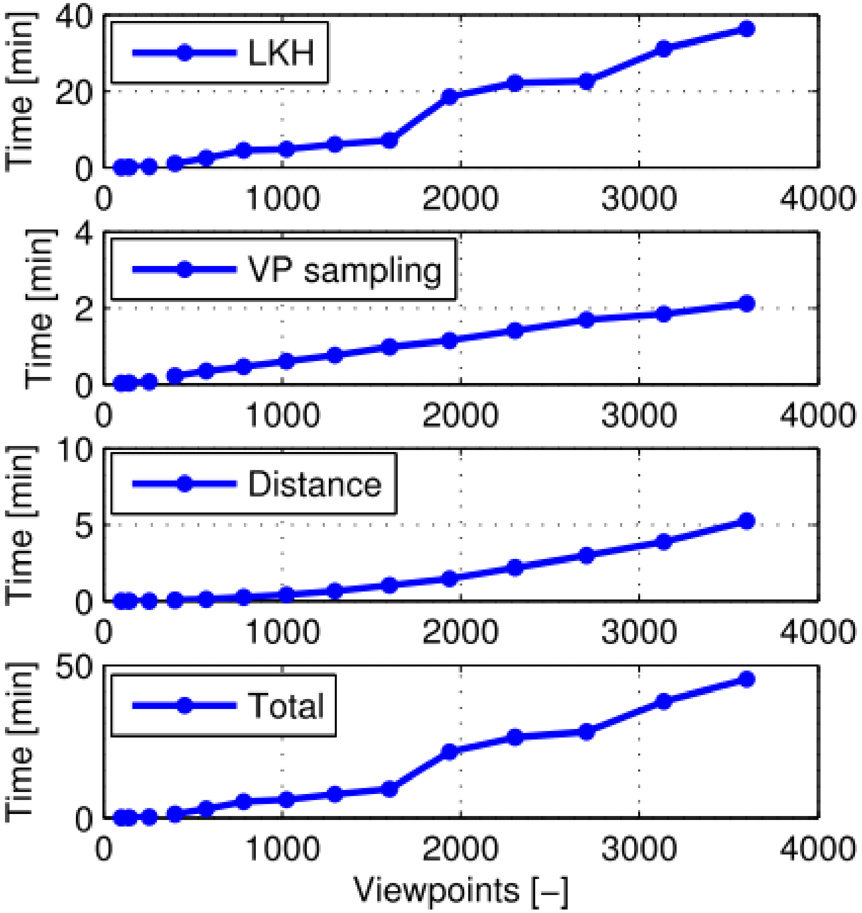
- The heading is determined according to

$$\min_{\psi^k} = \left(\psi_p^{k-1} - \psi^k\right)^2 / d_p + \left(\psi_s^{k-1} - \psi^k\right)^2 / d_s, \qquad \text{s.t.} \qquad \textbf{Visible}(g^k, \psi^k)$$

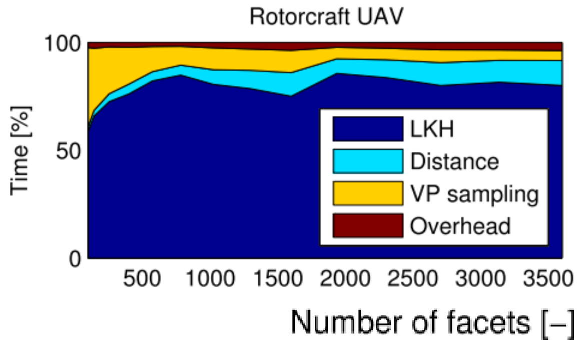
$\textbf{Visible}(g^k, \psi^k)$ means that from the given configuration, g^k and ψ^k, the whole triangle is visible. d_p and d_s are the Euclidean distances from g^k to g_p^{k-1} and g_s^{k-1} respectively.

Computational Analysis

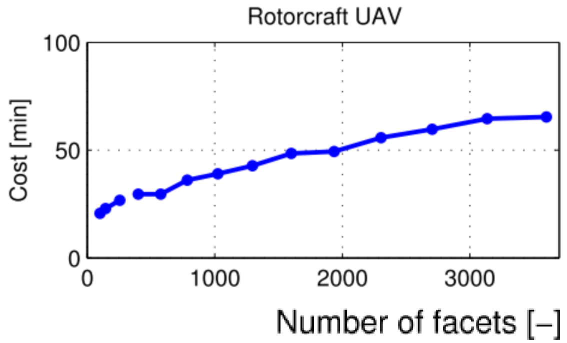
- To evaluate the capabilities, a simple scenario is used.



Facets	variable	incidence	30°
FoV	[70,70] °	Mounting pitch	25°
d_{min}	200m	d_{max}	200m
v_{max}	5m/s	ψ_{max}	0.5rad/s



(a) Relative time consumption



(b) Resolution dependent cost

The time complexity:
LKH: $O(N^{2.2})$
VP Sampling: $O(N)$
Distance compute.: $O(N^2)$

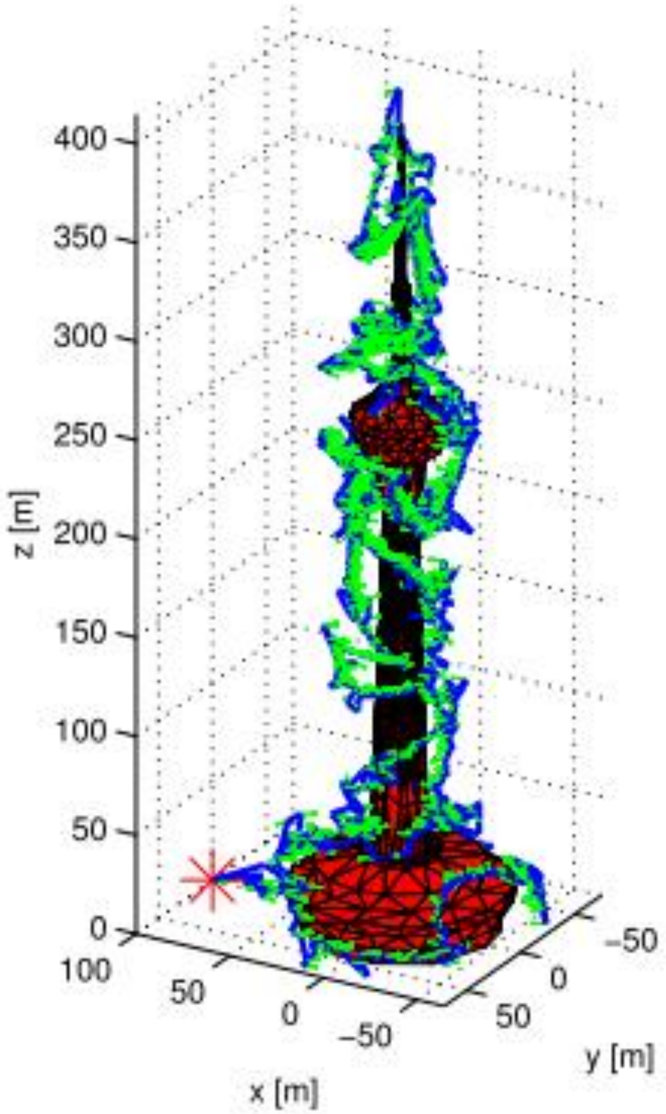
Evaluation Test - Simulation

- 405m Tower

N_{facets}	1701	$\angle$incidence	$30°$
FoV	$[120, 120]°$	Mouting pitch	$15°$
$d_{\min}$	10m	$d_{\max}$	25m
$v_{\max}$	2m/s	$\dot{\psi}_{\max}$	0.5rad/s

Large scale structure to be inspected:
The 405m high Central Radio & TV Tower in Beijing. The mesh used to compute the path contains 1701 triangular facets.
After a computation time of 92s the cost for the inspection is 2997.44s

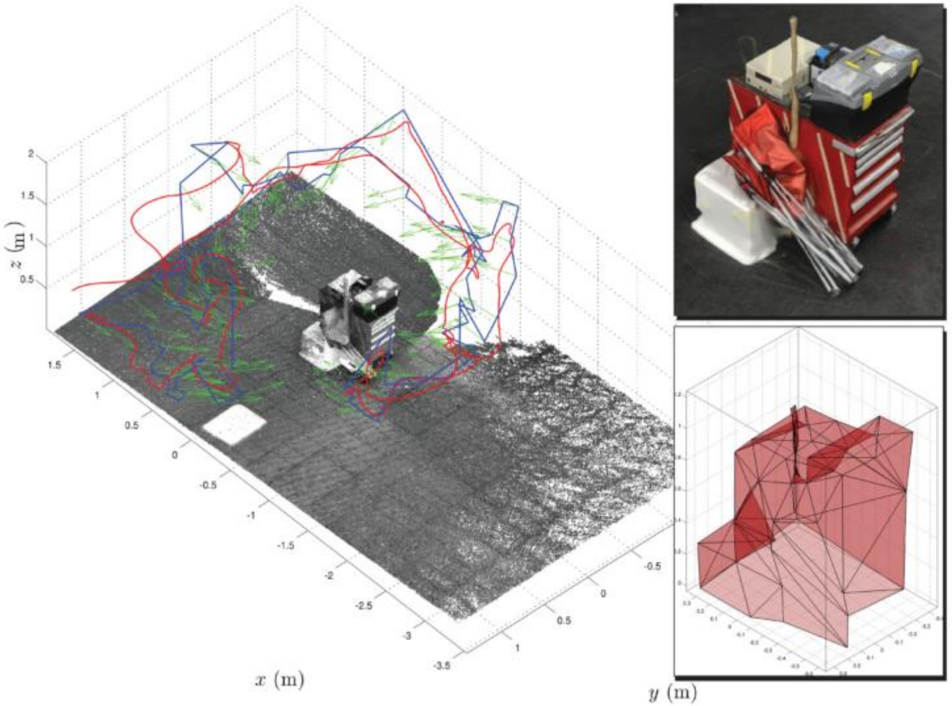
The red point denotes, start– and end–point of the inspection.



Evaluation Test – VTOL UAV

- Online: image processing
- Offline: 3D reconstruction (Image->Pix4D)
  Path planner
- Cost for the inspection: 151.44s

Visual-Inertial Sensor, ATOM CPU (Linux)



N_{facets}	106		
$\angle$incidence	$30°$	Bounding box	3x3x2.75m
FoV	$[60, 90]°$	Mouting pitch	$15°$
d_{min}	1m	d_{max}	3m
v_{max}	0.25m/s	$\dot{\psi}_{max}$	0.5rad/s

Summary & Conclusions

- A *practically–oriented* fast inspection path planning algorithm capable of computing efficient solutions for complex 3Dstructures represented by triangular meshes was presented.

- With the help of 3D reconstruction software, the recorded inspection data were post-processed to support the claim of finding full coverage paths and the point cloud datasets are released to enable evaluation of the inspection quality.

- **https://github.com/ethz-asl/StructuralInspectionPlanner**

URBAN ROBOTICS LAB

2.

Multi-layer Coverage Path Planner for Autonomous Structural Inspection of High-rise Structures

KAIST

URBAN ROBOTICS LAB

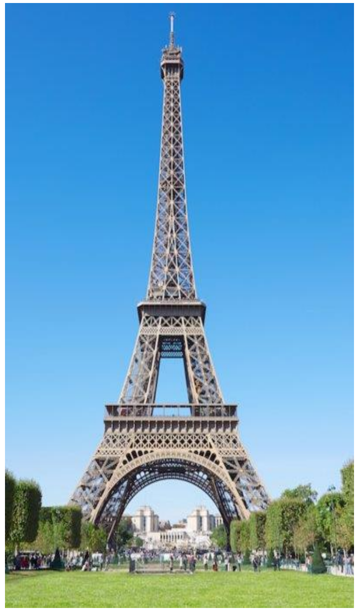
Intro.

- Structural inspection and maintenance of large structure is becoming significantly important.

Lotte World Tower, Seoul

Central TV Tower, Beijing

Oriental Pearl Tower, Shanghai

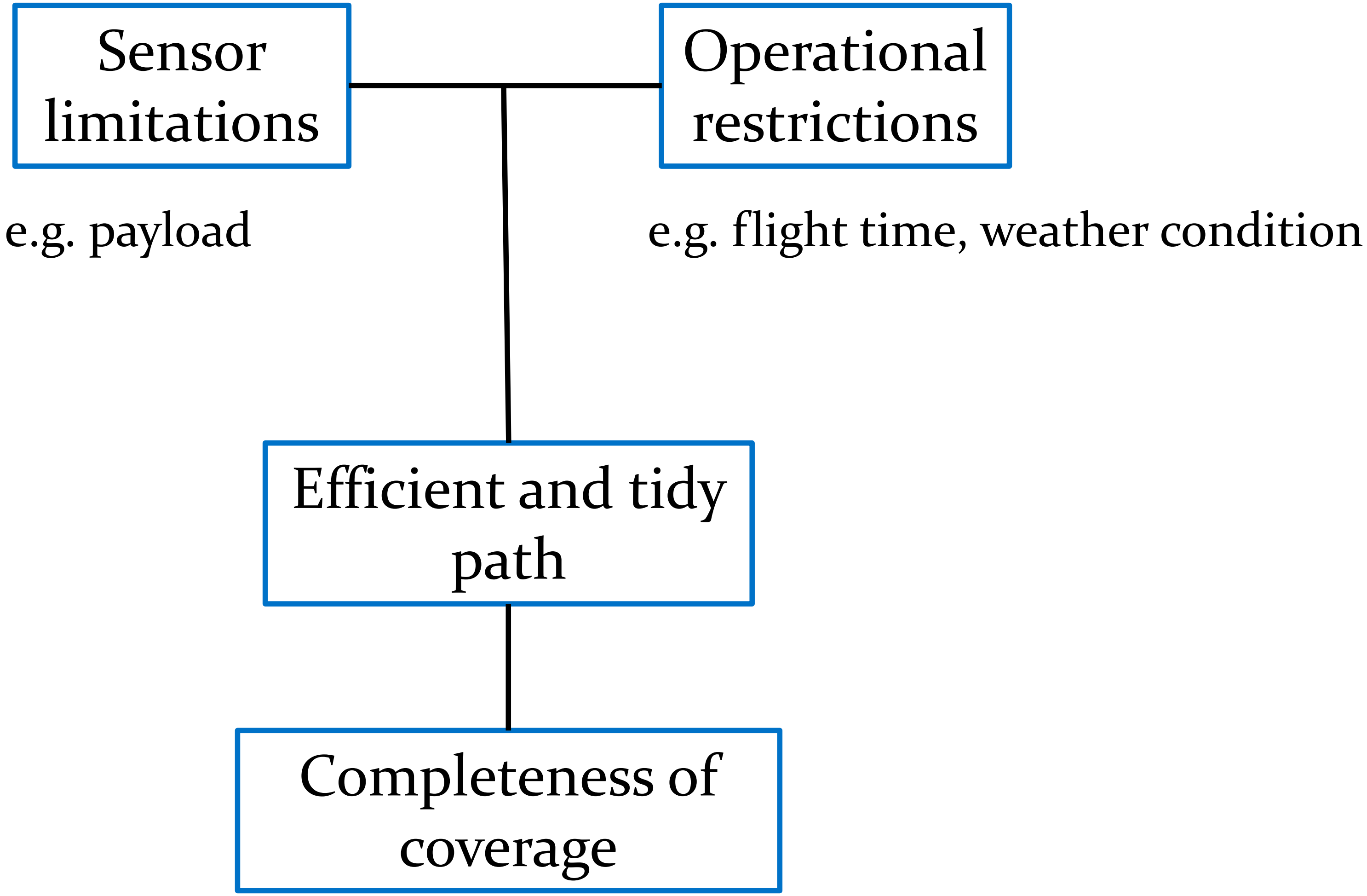


Eiffel Tower, Paris

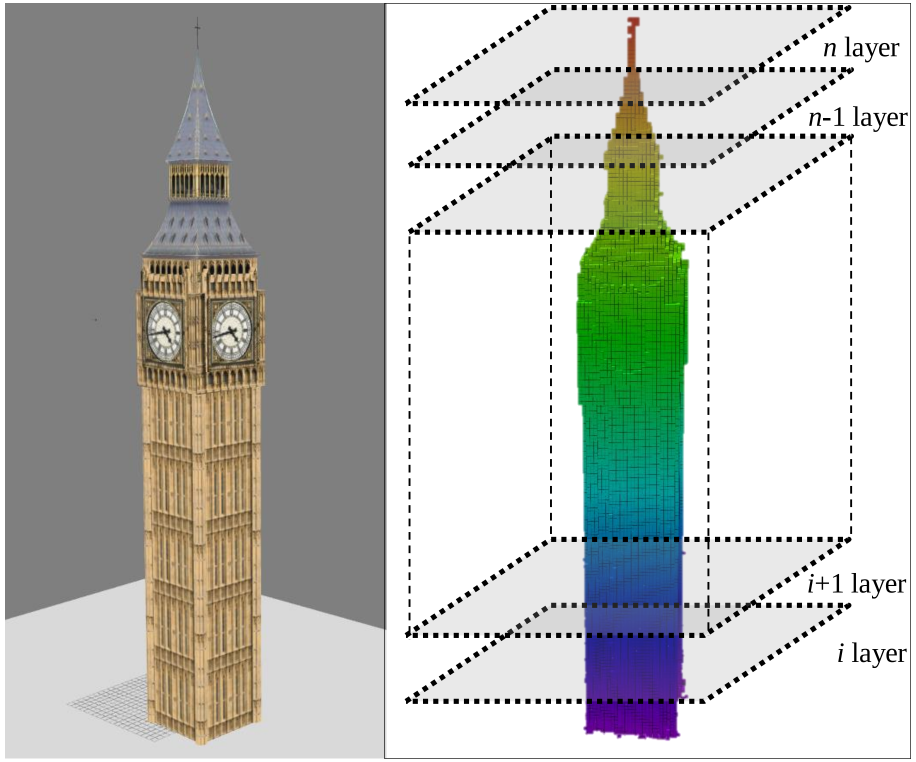
- Using UAV,

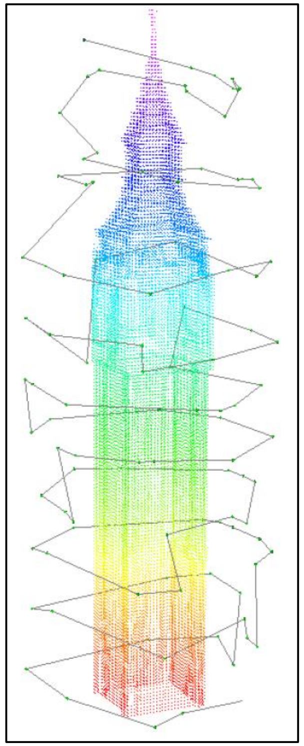
it is faster, safer, cheaper !

Intro.


Sensor limitations
Operational restrictions
e.g. payload
e.g. flight time, weather condition
Efficient and tidy path
Completeness of coverage

Contribution





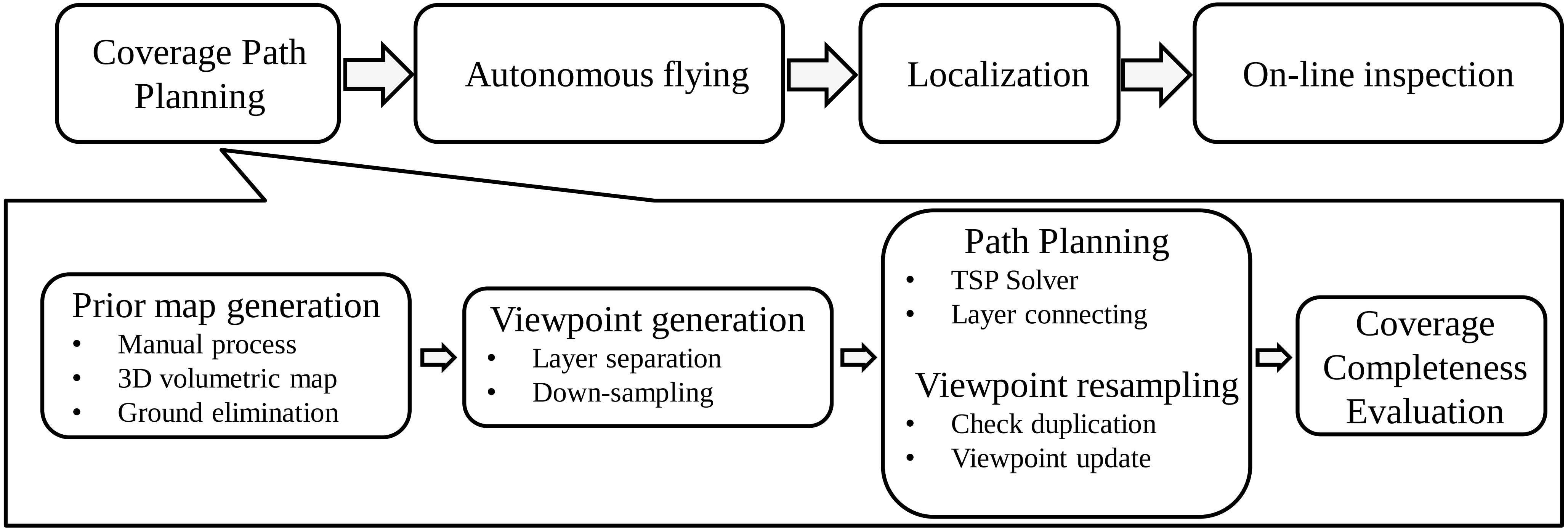
K: # of layer
n: # of viewpoint in each layer

Spiral path → Efficient and tidy

Computational complexity → $\mathcal{O}(N^{2.2}) \Rightarrow \mathcal{O}({n_1}^{2.2}) + \cdots + \mathcal{O}({n_K}^{2.2})$

$N = n_1 + n_2 + \cdots + n_K$

Methodology



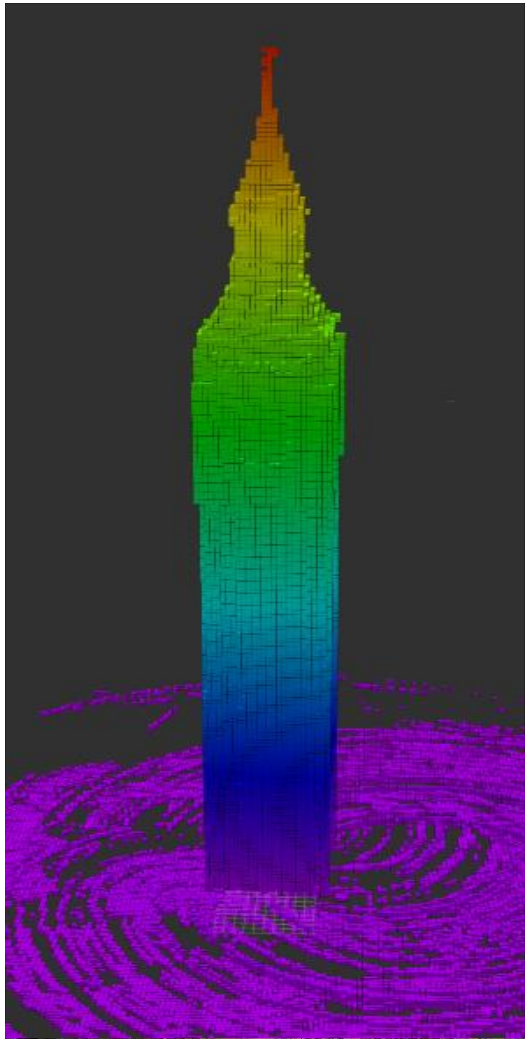
Methodology

Multi-Layer Coverage Path Planner

Input: DistToStruct, FOV, VoxelSize, StartPoint, NumOfLayers

```
Generating voxelized 3D map
Calculate a surface normal vector (n⃗_1, n⃗_2 ⋯, n⃗_N) of every center point (C_{1~k})
Divide the structure with K layers by height
while i < K do
    Sample initial viewpoint (v_1, v_2, …, v_N) at i-th layer
    Down-sample essential viewpoints (v̂_1, v̂_2, …, v̂_N)
    Solve TSP problem with LKH solver at i-th layer
    Update VPs in (i+1)-th layer by checking duplication
    Connect i layer and i+1 layer
    i ← i+1
```

Output: TourLength, CalcTime



Methodology

Multi-Layer Coverage Path Planner

Input: DistToStruct, FOV, VoxelSize, StartPoint, NumOfLayers

Generating voxelized 3D map

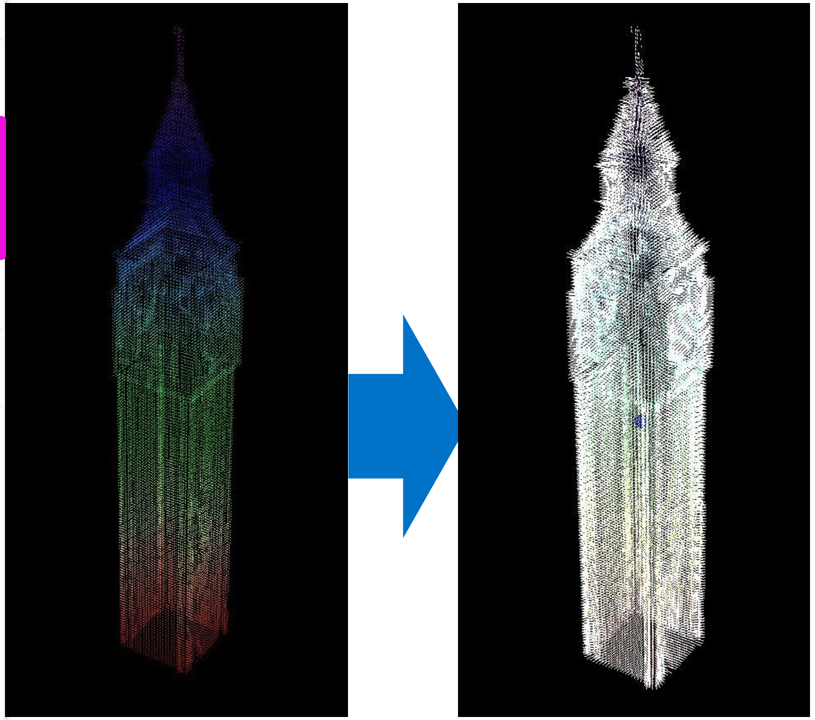
Calculate a surface normal vector $(\vec{n}_1, \vec{n}_2 \cdots, \vec{n}_N)$ of every center point $(C_{1\sim k})$

Divide the structure with K layers by height

while $i < K$ **do**

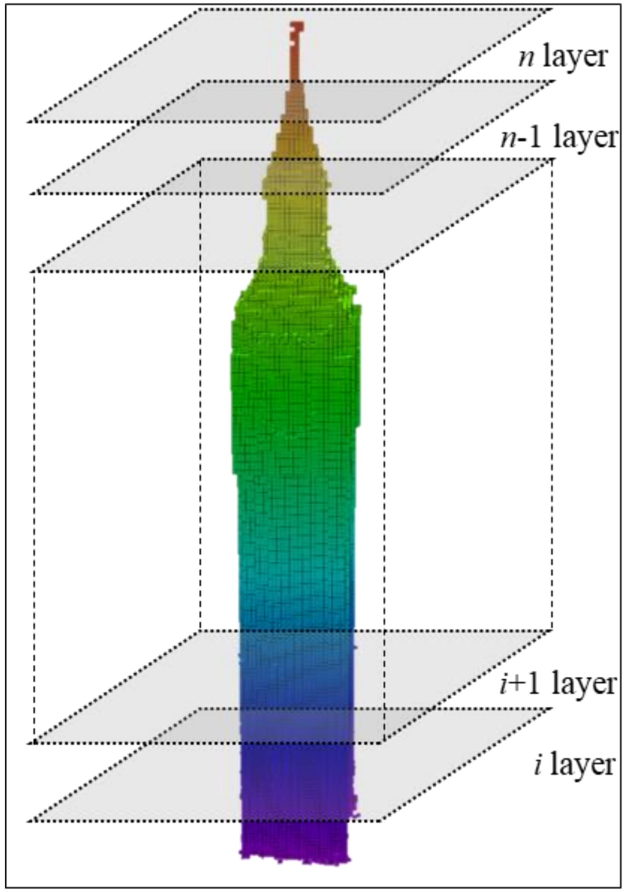
    Sample initial viewpoint $(v_1, v_2, \dots, v_N)$ at i-th layer
    Down-sample essential viewpoints $(\hat{v}_1, \hat{v}_2, \dots, \hat{v}_N)$
    Solve TSP problem with LKH solver at i-th layer
    Update VPs in (i+1)-th layer by checking duplication
    Connect i layer and i+1 layer
    $i \leftarrow i$+1

Output: TourLength, CalcTime



Methodology

Multi-Layer Coverage Path Planner
Input: DistToStruct, FOV, VoxelSize, StartPoint, NumOfLayers
1: Generating voxelized 3D map
2: Calculate a surface normal vector $(\vec{n}_1, \vec{n}_2 \cdots, \vec{n}_N)$ of every center point $(C_{1\sim k})$
3: Divide the structure with K layers by height
4: **while** $i < K$ **do**
5: Sample initial viewpoint $(v_1, v_2, \ldots, v_N)$ at i-th layer 6: Down-sample essential viewpoints $(\hat{v}_1, \hat{v}_2, \ldots, \hat{v}_N)$ 7: Solve TSP problem with LKH solver at i-th layer 8: Update VPs in (i+1)-th layer by checking duplication 9: Connect i layer and i+1 layer 10: $i \leftarrow i$+1
Output: TourLength, CalcTime



KAIST

URBAN ROBOTICS LAB

Methodology

Multi-Layer Coverage Path Planner

Input: DistToStruct, FOV, VoxelSize, StartPoint, NumOfLayers

Generating voxelized 3D map

Calculate a surface normal vector $(\vec{n}_1, \vec{n}_2 \cdots, \vec{n}_N)$ of every center point $(C_{1\sim k})$

Divide the structure with K layers by height

while $i < K$ **do**

  Sample initial viewpoint $(v_1, v_2, \ldots, v_N)$ at i-th layer

  Down-sample essential viewpoints $(\hat{v}_1, \hat{v}_2, \ldots, \hat{v}_N)$

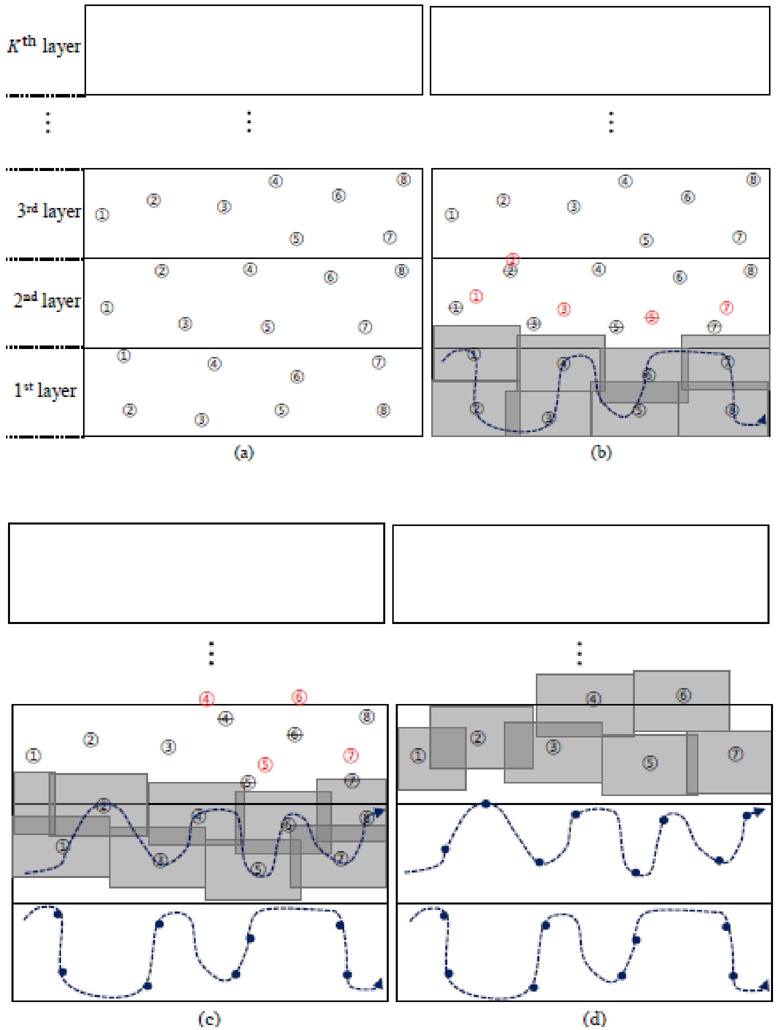
  Solve TSP problem with LKH solver at i-th layer

  Update VPs in $(i+1)$-th layer by checking duplication

  Connect i-th layer and $(i+1)$-th layer

  $i \leftarrow i+1$

Output: TourLength, CalcTime



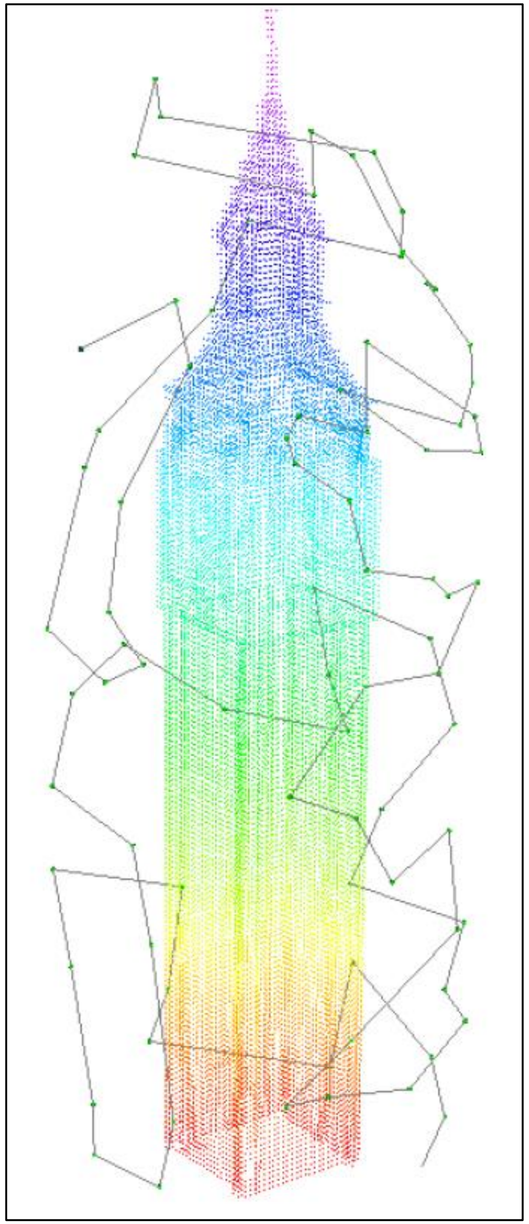
Experiment

Experiment

- No layer
  - Initial viewpoints : 3441
  - Selected viewpoints : 83

  - Computation time
    - Down sampling: 435.206 seconds
    - TSP: 0.840438 seconds
    - VP update: -
    - Total: 436.0464 seconds

  - Total distance: 3505.15m
  - Missed voxel: 311/19935

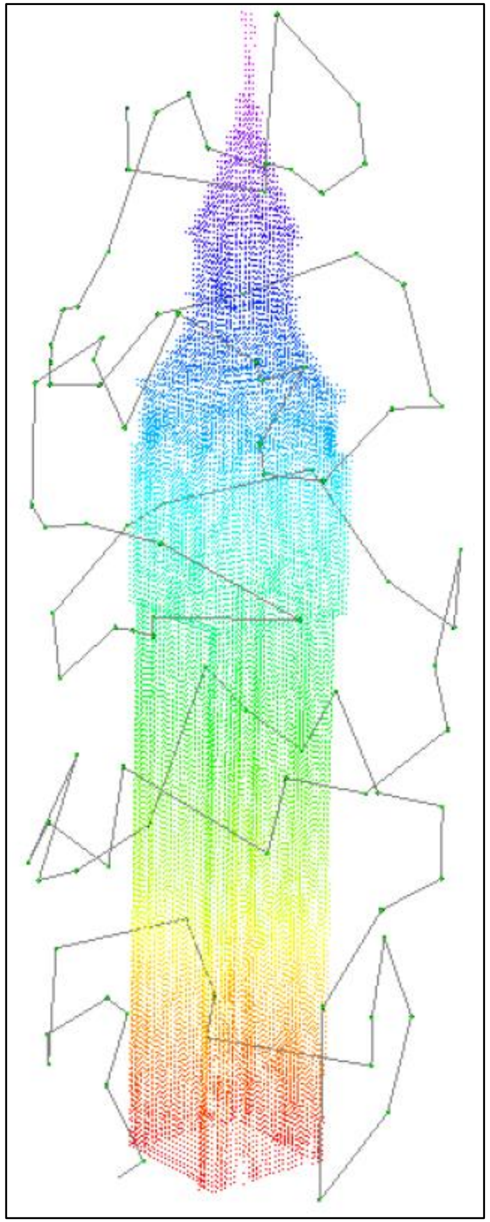


Experiment

- 5-layers (every 20m)
  - Initial viewpoints: 2885
  - Selected viewpoints: 95

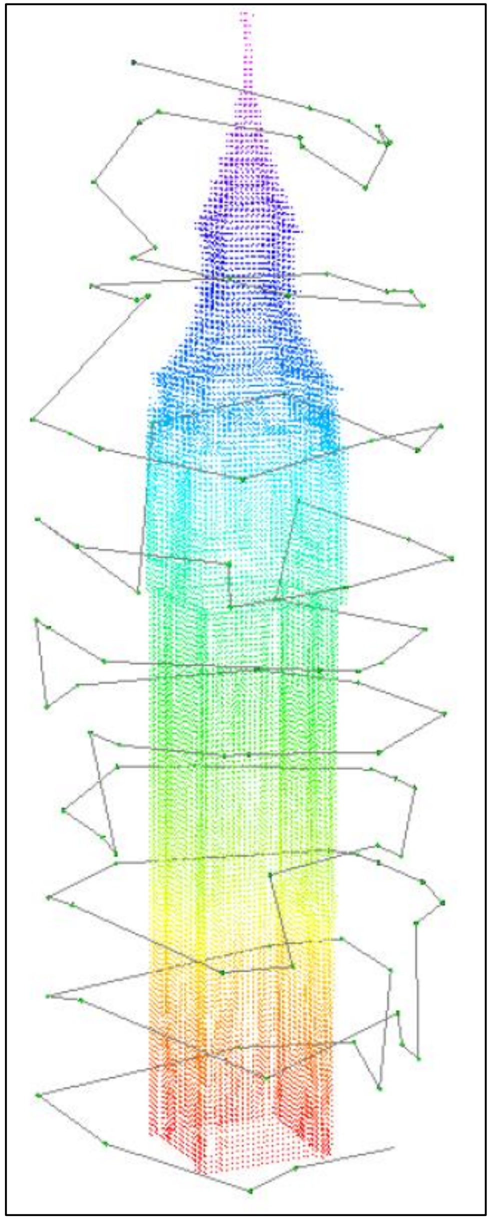
  - Computation time
    - Down sampling: 79.65832 seconds
    - TSP: 1.069598 seconds
    - VP update: 4.358884 seconds
    - Total: 85.0868 seconds

  - Total distance: 1943.623m
  - Missed voxel: 156/19935



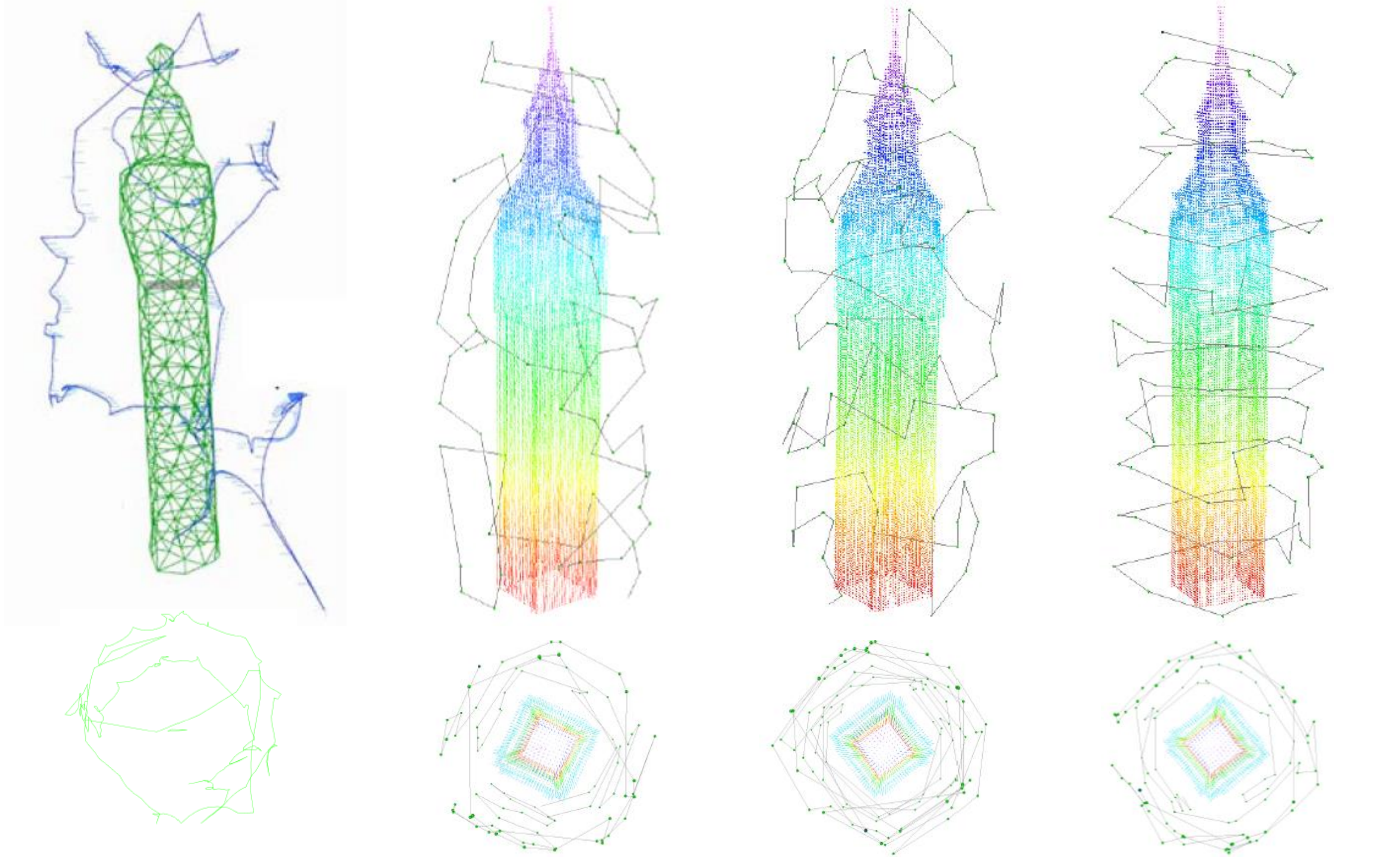
Experiment

- **12-layers (every 8m)**
  - Initial viewpoints : 1642
  - Selected viewpoints : 102
  - Computation time
    - Down sampling: 15.20832 seconds
    - TSP: 0.07194 seconds
    - VP update: 5.385357 seconds
    - Total: 20.66561 seconds
  - Total distance: 2165.702m
  - Missed voxel: 36/19935



Results

A. Bircher et al. [1]	No-Layer	5-Layers	12-Layers



Results

- Total Comparison

	SIPP [17]	No-Layer	5-Layer	12-layer
Dist. to target	10∼50	10	10	10
Num. of VP	526	83	95	102
Sampling time(s)	-	296.7	79.6	15.2
TSP time(s)	24.8	4.84	1.07	0.07
VP update time(s)	-	134.6	4.3	5.3
Total time(s)	$\approx$ 30	436.1	84.9	20.5
Tour length(m)	$\approx$ 2000	3505.1	1943.6	2165.7
Completeness(%) (missed voxel /total voxel)	-	98.4 (311 /19935)	99.2 (156 /19935)	99.8 (36 /19935)

-: not mentioned or not exist

[1] Bircher, Andreas, et al. "Structural inspection path planning via iterative viewpoint resampling with application to aerial robotics," *IEEE International Conference on Robotics and Automation (ICRA),* pp. 6423-6430, Seattle, USA, 2015.

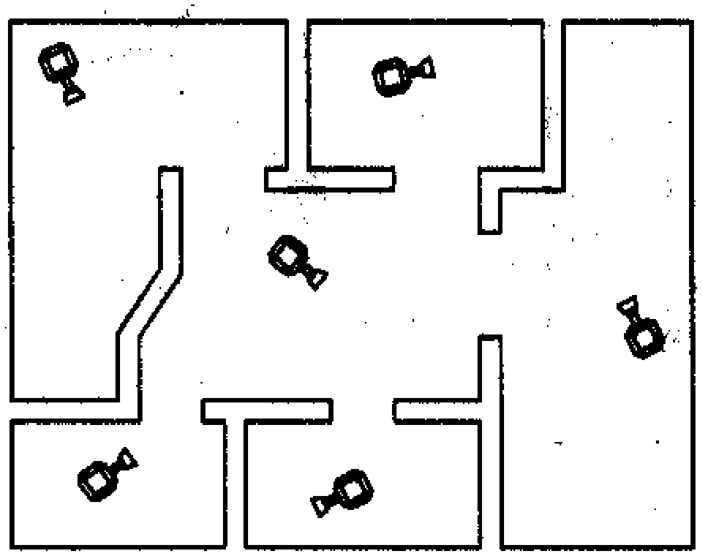
KAIST

URBAN ROBOTICS LAB

Appendix.1

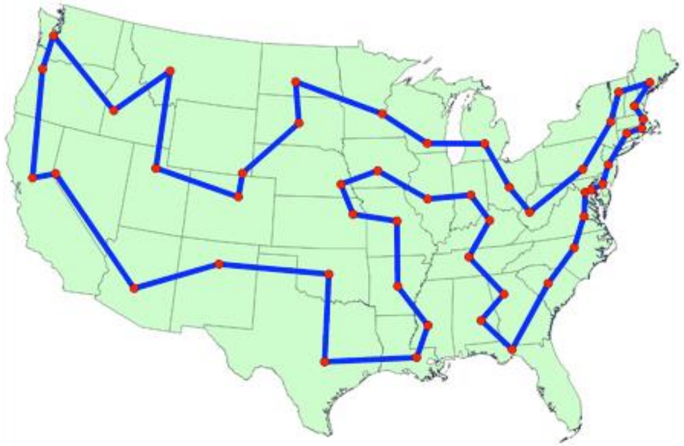
Art Gallery Problem (AGP)

Suppose you have an art gallery containing priceless paintings and sculptures. You would like it to be supervised by security guards, and you want to employ enough of them so that at any one time the guards can between them oversee the whole gallery. How many guards will you need?



Travelling Salesman Problem (TSP)

Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?



Appendix.2

Boundary Value Problem (or Solution)

- A boundary value problem has conditions specified at the extremes ("boundaries") of the independent variable in the equation

Example: Find a solution to the BVP problem

$$\frac{d^2y}{dx^2} - y = 0; \quad y(0) = 0, y(1) = 1 \text{ if we know}$$

$y(x) = c_1 e^x + c_2 e^{-x}$ *is a general solution to the differential equation.*

- whereas an initial value problem has all of the conditions specified at the same value of the independent variable (and that value is at the lower boundary of the domain, thus the term "initial" value).

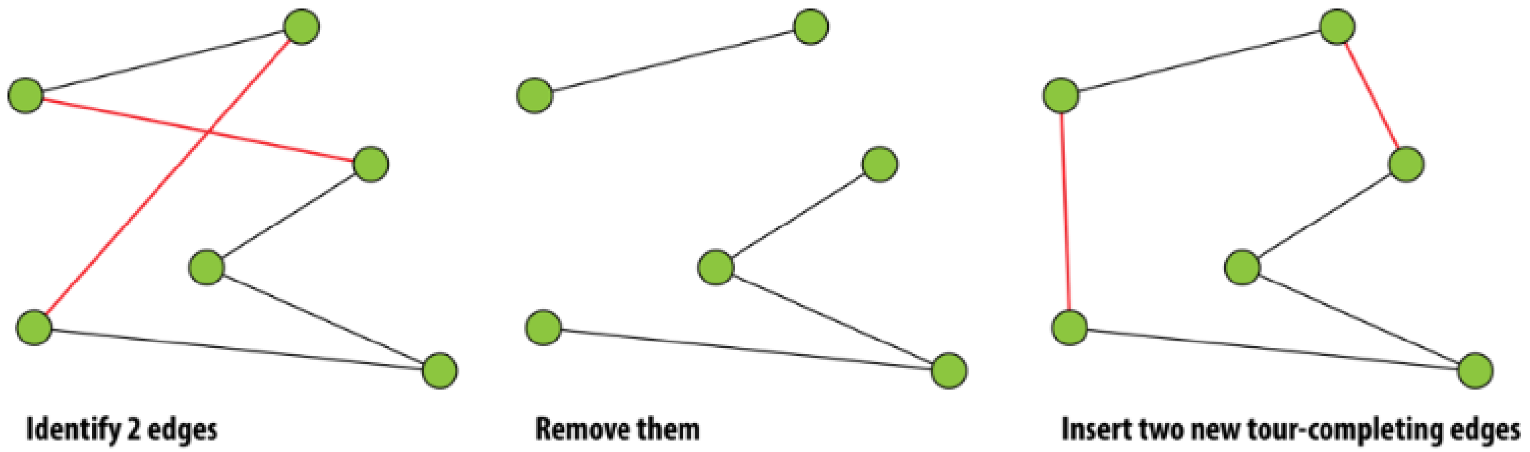
Example: Find a solution to the initial value problem

$$y'' + 4y = 0; \quad y(0) = 1; y'(\pi/2) = 2 \text{ if we know}$$

$y(x) = c_1 cos(2x) + c_2 sin(2x)$ *is a general solution to the differential equation.*

Appendix.3

- Lin-Kernighan-Helsgaun heuristic (LKH, K. Helsgaun, 1998)
  - LKH is an effective implementation of the Lin-Kernighan heuristic for solving the traveling salesman problem.
  - Even though the algorithm is approximate, optimal solutions are produced with an impressively high frequency.
  - Employ the concept of k-opt moves



Visualization of the k-moves process