Supplemental Material
TNS: Terrain Traversability Mapping and Navigation System for Autonomous Excavators

Tianrui Guan1*  Zhenpeng He1  Ruitao Song1  Dinesh Manocha2  Liangjun Zhang1
1 Robotics and Auto-Driving Laboratory, Baidu Research
2 University of Maryland, College Park

I. MORE DETAILS OF THE CWT DATASET

Our dataset is collected at a construction site while an excavator is navigating through the work area. We collect 3 videos that total approximately 30 minutes; 669 images of size $1920 \times 1080$ with pixel-wise annotation are included in our dataset. Please refer to this link for the access to the CWT dataset.

A. Dataset Details and Statistics

There are three video sequences collected on our excavator test site. In Figure 1, we show the class distribution breakdown for three sequences. The first video is collected after rain and consists of mostly water and muddy ground. The trenches caused by excavation and navigation can also be seen. The other two video sequences are captured on a sunny day in different scenarios. The videos are collected by a professional operator controlling the movement of the robot. The three videos are 268s, 668s, and 822s. We sample the camera stream every two seconds and annotate the images with ground truth labels, resulting in a total of 669 images after removing some redundant ones.

B. Benchmarks

We give several metrics and show the performance of several SOTA methods on the CWT dataset in Table I. $C$ denotes the set of all classes.

* Work done during an internship at Baidu RAL.

Fig. 3: Grid map output comparison: We show slope (left), step height (middle), and roughness (right) values in geometric traversability computations. All values are converted to the scale from 0 to 1. Slope value tends to have many small areas of peaks, while step height tends to have smoother values across a bigger region. On the other hand, roughness does not have many peak values, and many regions like hills are already captured by the previous two measurements.

$$mIoU = \frac{1}{c} \sum_{c=1}^{C} \frac{TP_{c}}{TP_{c} + FP_{c} + FN_{c}}$$

$$mAcc = \frac{1}{c} \sum_{c=1}^{C} \frac{TP_{c}}{TP_{c} + FN_{c}}$$

$$aAcc = \frac{\sum_{c=1}^{C} TP_{c}}{Numbers \ of \ All \ Pixels}$$
TABLE I: Performance of SOTA methods on the CWT dataset: We list several SOTA semantic segmentation methods and train the model with 240K iterations. * marks methods that do not converge well after 240K additional iterations.

<table>
<thead>
<tr>
<th>Year</th>
<th>Methods</th>
<th>Flat</th>
<th>Bumpy</th>
<th>Water</th>
<th>Rock</th>
<th>Mixed</th>
<th>Excavator</th>
<th>Obstacle</th>
<th>mIoU</th>
<th>mAcc</th>
</tr>
</thead>
<tbody>
<tr>
<td>2018</td>
<td>CGNet [10]</td>
<td>73.02</td>
<td>63.11</td>
<td>38.22</td>
<td>47.0</td>
<td>47.04</td>
<td>35.78</td>
<td></td>
<td>53.41</td>
<td>67.59</td>
</tr>
<tr>
<td>2019</td>
<td>Fast SCNN [6]</td>
<td>74.1</td>
<td>65.87</td>
<td>32.02</td>
<td>46.58</td>
<td>45.51</td>
<td>45.91</td>
<td></td>
<td>54.77</td>
<td>68.75</td>
</tr>
<tr>
<td>2019</td>
<td>Fast FCN [9]</td>
<td>71.96</td>
<td>61.23</td>
<td>35.61</td>
<td>46.42</td>
<td>41.02</td>
<td>37.22</td>
<td></td>
<td>54.37</td>
<td>67.05</td>
</tr>
<tr>
<td>2021</td>
<td>BiSeNetV2 [12]</td>
<td>76.49</td>
<td>69.65</td>
<td>38.33</td>
<td>71.44</td>
<td>46.42</td>
<td>41.02</td>
<td></td>
<td>54.37</td>
<td>67.05</td>
</tr>
<tr>
<td>2021</td>
<td>SETR* [13]</td>
<td>54.24</td>
<td>49.67</td>
<td>4.07</td>
<td>25.23</td>
<td>6.03</td>
<td>0.0</td>
<td></td>
<td>19.91</td>
<td>30.61</td>
</tr>
<tr>
<td>2021</td>
<td>DPT* [7]</td>
<td>59.45</td>
<td>53.75</td>
<td>23.78</td>
<td>33.69</td>
<td>26.0</td>
<td>0.0</td>
<td></td>
<td>29.02</td>
<td>47.65</td>
</tr>
<tr>
<td>2021</td>
<td>Segformer [11]</td>
<td>73.44</td>
<td>64.47</td>
<td>39.62</td>
<td>70.29</td>
<td>43.81</td>
<td>30.48</td>
<td></td>
<td>50.6</td>
<td>64.29</td>
</tr>
</tbody>
</table>

II. MORE DETAILS OF TNS

A. Roughness in Geometric Traversability

In many existing works [1, 8] for geometric traversability or danger value calculation, a roughness score is calculated as a factor of terrain traversability.

**Roughness Estimation:** The terrain roughness $r$ is calculated as the standard deviation of the terrain height values to the fitting plane. The distance $d$ from the center point $p$ of the grid to the fitting plane of $k$ neighboring grids is calculated as:

$$d = \frac{\bar{p} \cdot \vec{n}}{||\vec{n}||}$$

where $\vec{n}$ is the surface normal vector of the fitting plane and $\bar{p}$ is a point in the plane. Finally, the roughness estimation of the grid $g$ can be computed as:

$$r = \sqrt{\frac{1}{k} \sum_{i=1}^{k} (d_i)^2}$$

However, roughness is not a good measurement in unstructured environments, especially in our situation. During the design of our method, we discover that the roughness measurement is either random or, in some regions, the distribution of roughness resembles that of slope or step height, except with lower peak values. Eventually, the roughness score did not impact the results too much. We demonstrate such similarity in Figure 3.

III. MORE DETAILS OF TNS-BASED PLANNING

In this section, we add more details of our planning method and experimentation.

A. A* and Hybrid A*

A* [3] search can be seen as an improvement of Dijkstra’s search. Dijkstra calculates the cost to start $g(x)$ of each vertex to determine the next vertex to be expanded. A* search enhances the algorithm by using heuristic cost $h(x)$, allowing faster convergence under certain conditions, while still ensuring its optimality [4]. The heuristic cost $h(x)$ is the cost to goal based on a heuristic estimate of the cost from state $x$ to the goal state $x_{goal}$, since the actual cost $g(x)$ is the path that has been actually traversed. The total cost is thus

$$f(x) = g(x) + h(x)$$

by which the way points will be sorted. A standard heuristic estimate function is the Euclidean distance for two-dimensional problems.

---

Fig. 4: **Left:** A* associates costs with centers of cells and only visits states that correspond to grid-cell centers. **Right:** Hybrid A* associates a continuous state with each cell, and the score of the cell is the cost of its associated continuous state. A* path can always move to the center of the adjacent node, while in hybrid A*, we consider the actual motion constraints of the object, so the red dot does not appear in the grid center.

Fig. 5: **Comparison between original and improved hybrid A* planners.** Our planning method has more flexibility, including running over small obstacles between two tracks based on our traversability map.

The hybrid A*[2, 5] algorithm is proposed for path planning of nonholonomic robots. In A*, we do not consider the direction of the moving object, and we do not consider the actual movement of the object. However, in hybrid A*, we need to consider the constraint of the robot motion model. In Figure 4 we use the red dot to indicate the possible position of the robot. The differences between the two algorithms are shown in Table II and we also provide pseudo-code in Alg. 1.
Algorithm 1: Hybrid A* Search

Input: Start state: $x_s$; Goal state: $x_g$
Output: Valid path between $x_s$ and $x_g$

begin
  $O = \emptyset$ // Initialize Open set
  $C = \emptyset$ // Initialize Close set
  $f(x_s) = g(x_s) + h(x)$ // Update cost of $x_s$
  according to the cost function. $g(x)$ is the actual
  cost and $h(x)$ is the heuristic cost.
  $O.push(x_s)$
  while $O$ not empty do
    $x \leftarrow O.popMin()$ // $O.popMin()$ return the
    node with the lowest cost in $O$
    if $x == x_g$ then
      return path // Trace the parent node from
      the end point $x$, until it reaches the
      starting point, return to the result path
      found.
    else
      $C.push(x)$
      for each $n \in \text{neig}(x)$ do
        // Go through all collision-free
        neighbors of $x$ according to the
        kinematic model
        if $n \notin O$ then
          $f(n) = n.updateCost()$
          $O.push(n)$
      return null // Can not find a valid path
end

TABLE III: Success rate of planning in each scenario.
"Difficult Terrain" means the excavator must traverse through
or navigate around a rough region or water. "Obstacles" means
there are obstacles in the environment.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Difficult Terrain</th>
<th>Obstacles</th>
<th>Geometric Method (%)</th>
<th>TNS (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 1</td>
<td>✓</td>
<td>✓</td>
<td>40</td>
<td>60</td>
</tr>
<tr>
<td>Case 2</td>
<td>✓</td>
<td>✓</td>
<td>16.67</td>
<td>71.43</td>
</tr>
<tr>
<td>Case 3</td>
<td>✓</td>
<td>✓</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>Case 4</td>
<td>✓</td>
<td>✓</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td>Case 5</td>
<td>✓</td>
<td>✓</td>
<td>20</td>
<td>100</td>
</tr>
<tr>
<td>Case 6</td>
<td>✓</td>
<td>✓</td>
<td>40</td>
<td>80</td>
</tr>
<tr>
<td>Case 7</td>
<td>✓</td>
<td>✓</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>Case 8</td>
<td>✓</td>
<td>✓</td>
<td>20</td>
<td>60</td>
</tr>
<tr>
<td>Case 9</td>
<td>✓</td>
<td>✓</td>
<td>20</td>
<td>60</td>
</tr>
<tr>
<td>Overall</td>
<td>✓</td>
<td>✓</td>
<td>33.3</td>
<td>82.6</td>
</tr>
</tbody>
</table>

B. Experiment Details of Offline Planning

We compare the success rate of the planner using output
traversability maps from the geometric-only method [1]
and the proposed TNS. We use 9 different scenarios, and each
scenario is tested with the same starting points and random
goal position with more than 10 trials. We list the details of
all scenarios in Table III.

C. Failure Cases of Hybrid A* in Our Applications

We show some planning scenarios where traditional Hybrid
A* would fail in Fig 5.
REFERENCES


[13] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiutian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip H.S. Torr, and Li Zhang. Rethinking semantic segmentation from a sequence-to-