DEVELOPMENT OF A SOFTWARE TOOL TO IMPLEMENT A VISION-BASED NAVIGATION METHOD IN LOW-TEXTURED ENVIRONMENTS

Selection of Horizon Line Segmentation Techniques in Non-Urban Environments

April 1, 2025

1 Selection

For the selection of skyline segmentation techniques in non-urban environments, the following works were chosen:

[**Alpha — Status:** in Progress] In [1], a robust, classical approach to skyline detection is proposed using a dynamic programming (DP) algorithm. After detecting edges on the image's luminance channel, a multi-stage graph is built where columns represent stages and edge pixels are nodes. The Dynamic Programming (DP) search finds the lowest-cost path across the image, accounting for gaps and vertical smoothness. This method effectively extracts continuous skyline curves, even under challenging conditions like clouds or weak contrast. Considering that, and according to this work, the dynamic programming (DP)-based edge linking algorithm is used to extract the skyline robustly, the work plan for its implementation is as follows:

1. **Edge Detection**
   - Apply a standard edge detector (Canny and Sobel).
   - Apply thresholding (Otsu or Conditional Probability method) to produce a binary edge map.

2. **Multi-stage Graph Construction**
   - The image columns are treated as stages; edge pixels become graph nodes.
   - Node cost favors higher (upper) positions in the image.
   - Edge cost penalizes large vertical jumps between adjacent columns.

3. **Dynamic Programming Search**
   - Finds the lowest-cost path from left to right, allowing for small skyline breaks (gap tolerance tog).
   - Dummy nodes and penalty costs are added when edge continuity is missing.

Resources:

- Canny and Sobel edge detectors can be applied from Open CV python libraries, as well as the filtering and preprocessing steps.
- Dynamic programming is a well-known already documented algorithm. Plenty of documentation of its implementation through the internet (e.g, What is a Multistage graph in Python?). Dictionaries could replace matrices in order to assign a set of metadata to an attribute (node of the graph).

[**Beta - Status:** pending] Regarding [2], a complete system for skyline detection in glacial environments is presented, designed to handle low-contrast and white-out conditions. The method combines Canny edge detection with adaptive thresholding, followed by line simplification using the Ramer–Douglas-Peucker algorithm. Candidate segments are scored using visual heuristics like length, color cues, and proximity to the expected horizon. A greedy graph search then connects the best segments to form the skyline, which emphasizes robustness in harsh visual environments. The work plan for its implementation is as follows:

1. **Edge Detection**
   - Uses Canny edge detector with adaptive thresholding to extract dominant edges.

2. **Line Segmentation**
   - Applies the Ramer–Douglas–Peucker algorithm to simplify edge segments into piecewise linear lines.

3. **Heuristic-Based Scoring**
   Each candidate line segment is scored based on:
   - **Length** (longer = likely to be a skyline).
   - **Color similarity below the segment** to the known ground region.
   - **Color difference above the segment** from the ground region.
   - **Amount of "snow-colored" pixels** below the line.
   - **Distance from expected horizon** (if a pose estimate is available).

4. **Greedy Graph Search**
   - Starts from the best scoring segment and greedily connects neighboring segments. - Segments with strong scores can redirect the path; weak ones reinforce continuity.

Resources:

- Canny can be applied from Open CV python library, as well as the filtering and preprocessing steps.
- Ramer–Douglas-Peucker algorithm purpose is to simplify the number of points needed for a curve by approximating with a collection of points. There is even a library already made for this in Python

---

[**Gamma - Status: pending**] The approach in [3] introduces two complementary methods for horizon line detection—an edge-based method and a classification-based (edge-less) method—along with a fusion strategy that combines both. The edge-based method extracts Maximally Stable Extremal Edges (MSEE), classifies them using SIFT–HOG features with an SVM, and applies Dynamic Programming to trace the skyline. The edge-less method uses patch-based classification (SVM or CNN) to create a Dense Classifier Score Image (DCSI), which also guides a Dynamic Programming search. The fusion method blends both nodal scores to enhance robustness, particularly in challenging scenes like fog or partial occlusions. For this work, the use of **Support Vector Machines (SVMs)** is justified primarily due to their strong performance with a small training dataset and their effectiveness with the chosen hand-crafted features (specifically, SIFT–HOG combinations) for edge classification. This Machine Learning method works well with limited training data (only 9 training images used), is less computationally intensive compared to deeper neural networks, and has compatibility with the proposed dynamic programming framework. The work plan for its implementation is as follows:

A. Edge-based Method:

1. **Edge Detection**
   - Uses Maximally Stable Extremal Edges (MSEE) extracted from multi-thresholded Canny edges.
2. **Classification**
   - Edge pixels are classified using SIFT–HOG features and a trained SVM.
3. **Graph Search**
   - Constructs a multi-stage graph using only positively classified edges.
   - Applies Dynamic Programming to find the shortest path from left to right (the skyline).

B. Edge-less Method (Dense Classification):

1. Trains a classifier (SVM or CNN) using 16×16 image patches labeled as horizon or non-horizon.
2. Generates a Dense Classifier Score Image (DCSI): each pixel is scored by its likelihood of being on the horizon.
3. Applies Dynamic Programming to find the optimal skyline path over the DCSI.
4. This method is robust to edge gaps and image blur.

C. Fusion Method:

1. Combines the edge-based and edge-less nodal scores in the DP graph using a weighted average.
2. Boosts performance in difficult scenes (e.g., fog, partial horizon).

Resources:

- Canny can be applied from Open CV python library, as well as the filtering and preprocessing steps.
- MSER (Maximally Stable Extremal Regions) algorithm has several references on how to apply it, among them: MSER (Maximally Stable Extremal Regions).
- As long as the patent for SIFT already expired, OpenSIFT is an already made library for this. However, built-in ORB in ROS can be used.
- The implementation for a Support Vector Machine (SVM) as a classifier is possible if using the already made Scikit-learn library. The type of SVM needs yet to be addressed.

References

[1] W.-N. Lie, T. C.-I. Lin, T.-C. Lin, and K.-S. Hung, "A robust dynamic programming algorithm to extract skyline in images for navigation," *Pattern Recognition Letters*, vol. 26, no. 2, pp. 221–230, 2005. DOI: https://doi.org/10.1016/j.patrec.2004.08.021.

[2] S. Williams, "Visual arctic navigation: Techniques for autonomous agents in glacial environments," Ph.D. dissertation, Georgia Institute of Technology, 2011.

[3] T. Ahmad, G. Bebis, M. Nicolescu, A. Nefian, and T. Fong, "Horizon line detection using supervised learning and edge cues," *Computer Vision and Image Understanding*, vol. 191, p. 102 879, 2020. DOI: https://doi.org/10.1016/j.cviu.2019.102879.