1 Introduction

Canal shoreline detection is a necessity for autonomous navigation in canals. Such detection can be achieved through expensive LIDAR, however such equipment is unsuitable for environmental projects where funding might be limited. It would be more preferable to have a small, light, inexpensive sensor, such as camera to detect the canal shoreline. There are several challenges presented in shoreline detection using computer vision. As mentioned in Supreeth Achar, and Pascal Mettes, there is a large spatial and temporal variations in appearance of water in the presence of nearby structures and with reflections from the sky. Such variation makes traditional feature recognition fail, and this is usually overcome through machine learning, Supreeth Achar, . The disadvantage of machine learning is the difficulty of using it in real time environment due to performances. Another approach used is the texture recognition approach. The approach taken by this paper is the dynamic motion approach. In this paper we propose a combination of using brightness constancy assumption and dynamic motion analysis for canal shoreline detection.

2 Related Work

3 Method

This section describes the method used to detect the canal shoreline using both brightness constancy assumption and the dynamic motion of water.

3.1 Features Tracking and Initialization

A set amount of k trackers is initially initialized using the Shi-Tomasi corner detection algorithm that detects good features to track. Due to the strong corners presented in the canal shorelines, several features will be initiated on the shoreline. Then a pyramidal implementation of Lucas-Kanade optical flow algorithm is used to displace the tracker across the subsequent n frames. Due to the brightness constancy assumption of optical flow, trackers on water would be more likely to lost, leaving the remaining trackers on the canals. However, there will be trackers that remain on the water, this trackers can be detected through the inherent difference between the dynamic motion of water and land. This detection is done by calculating the entropy and the inter-tracker dissimilarity of the tracker as proposed by Pedro Santana.

3.2 Entropy Calculation

The equation of entropy is defined as:

$$H(p) = \frac{\log(\frac{L(p)}{d(p)})}{\log(n-1)} \cdot d_\theta(p) \tag{1}$$

where p is defined as a vector of n positions relative to the tracker's initial position

$$p \equiv (x_0, \ldots, x_n) \tag{2}$$

and where $L(p)$ is defined as the length of the trajectory.

$$L(p) = \sum_{i=0}^{n-1} \|x_i - x_{i-1}\| \tag{3}$$

furthermore $d(p)$ is defined the diameter of the minimum circle encompassing the trajectory; $d_\theta(p)$ is the scaled version of $d(p)$ so that the range of $d_\theta(\cdot)$ value across the set of trackers, $\{d_\theta(x), \forall x \in \theta\}$, is $[0, 1]$.

3.3 Dissimilarity Calculation

Tracker's movement can also be influenced by the camera motion. As such, complex movement of camera can be translated to high entropy value of the tracker. This challenge can be considerably mitigated by assuming that the environment is piecewise planar and, thus camera motion is felt similarly in the neighbour trackers.

Formally, dissimilarity between a given tracker p and a given tracker q is defined by their Euclidean metric:

$$d(p, q) = \sum_{i=0}^{n} \|x_i - y_i\| \tag{4}$$

where y_i represents a 2-D position in the trajectory executed by q:

$$q \equiv (y_0 \ldots y_i) \tag{5}$$

Using the dissimilarity metric the dissimilarity between the trajectory of a given tracker p and the trajectories of its neighbour trackers:

$$D(p) = \frac{1}{\Phi(p)} \sum_{\forall q \in \Phi(p)} d(p, q) \tag{6}$$

3.4 Fusing trackers values

The entropy and dissimilarity value is tracked the following way:

$$F(p) = H_\theta(p) \cdot D_\theta(p) \tag{7}$$

where the $H_\theta(p)$ and $D_\theta(p)$ are the normalized version of $H_(p)$ and $D_(p)$ respectively. The normalization procedure is the same as defined above.

3.5 Finding Threshold

Since there is a significant difference between the fusion value of water tracker and fusion value of land tracker, it is possible to use 1D segmentation algorithm to determine the threshold value to differentiate land from water. Using an 1D segmentation algorithm, the Jenks Natural Break algorithm, the threshold is determined.

3.6 Finding Shoreline