DISASTER SCENE ANALYSIS AND SIMULATION USING LASER RANGE IMAGES

Atsushi NAKAZAWA, Cybermedia Center, Osaka University.
1-32, Machikaneyama, Toyonaka, Osaka 560-0043, JAPAN
Email: nakazawa@cmc.osaka-u.ac.jp

Kensuke YASUFUKU, Cybermedia Center, Osaka University.
1-32, Machikaneyama, Toyonaka, Osaka 560-0043, JAPAN
Email: yasufuku@cmc.osaka-u.ac.jp

Keywords: Laser Range Image, Image Recognition, Disaster Simulation, 3D

SUMMARY

Laser range images are used for varieties of tasks such as surveillance, robot vision or outdoor scene modeling. Our project aims to develop an algorithm to recognize outdoor disaster scenes by using range sensor images for robot’s navigations, creating disaster maps and evaluation plans. We use a mobile robot which has a line scan laser range sensor and a rotation stage. The sensor is able to scan a range data in a vertical direction and surrounding range image can be taken by rotating the sensor by rotation stage. The obtained range images are segmented into semantic regions such as people, cars or buildings. This segmentation result is used for following application tasks.

1) Creation of 3D digital maps onto Google Map images
By aligning captured range images, we can automatically create 3D maps of the large environment. Here, we used Google Map data for the ‘ground-truth’ map. The captured and segmented range images are matched to the Google Map image according to the semantic segmentation results and we can create 3D maps of the large environment.

2) Simulation of evacuations
According to the large environmental map and semantic segmentation results, we can perform the evacuation simulations by using multi-agent based evacuation simulation techniques. By combining the recognition algorithm and digital mapping technique, users can easily grasp the disaster situation and find the optimal way of evacuation.

3) Recognition of the disaster buildings
We applied machine learning techniques to the range images to find disaster building. Namely, we prepared miniature buildings of normal condition and collapsed condition and the image features from them are learned. The learning results are used to find the collapsed structured of the actual range images.

1. INTRODUCTION

Laser range images are used for varieties of tasks such as surveillance, robot vision or outdoor scene modeling. Our project aims to develop an algorithm to recognize outdoor disaster scenes by using range sensor images for robot’s navigations, creating disaster maps and evaluation plans. We use a mobile robot which has a line scan laser range sensor and a rotation stage. The sensor is able to scan a range data in a vertical direction and surrounding range image can be taken by rotating the sensor by rotation stage. The obtained range images are segmented into semantic regions such as people, cars or buildings. This segmentation result is used for following application tasks.

1) Creation of 3D digital maps onto Google Map images

2. RECOGNITION OF RANGE IMAGES

Object classification of input images is necessary for applications including robot navigation and automation, in particular with respect to path planning. To achieve robust object classification, we propose the idea of an object feature which represents a distribution of neighboring points around a target point. In addition, rather than processing raw points, we reconstruct polygons from the point data, introducing connectivity to the points. With these ideas, we can refine the Markov Random Field (MRF) calculation with more relevant information with regards to determining “related points”. The algorithm was tested against five outdoor scenes and
provided accurate classification even in the presence of many classes of interest.

2.1 RANGE IMAGE DATA
Input scenes are captured as laser range data taken by a SICK laser rotated 360 degree. The laser rotates up and down along scan lines and returns the distance (range) to any surface intersecting this scan line. As the laser is rotated, scan lines are taken at increasing rotation angles until the range data for the entire surroundings is captured (Figure 1). The laser rotation angle \( f \), scan line angle from the horizontal \( q \), and range \( r \) can be converted into Cartesian coordinates \( (x;y;z) \) using a standard polar-Cartesian conversion. While this input image is made up entirely of points (leading to the name point-cloud), it is possible to roughly reconstruct surface polygons from the points by connecting nearby points in adjacent scan-lines. The result of this is an input scene with points \( p_i \in P \) and polygons \( x_i \in X \). The use of polygons provides the advantage over raw points that there is connectivity information that can be applied to edge potentials in the Markov random field.

![Figure 1 Our scanning robot (left) and an captured range image](image)

2.2 LOCAL FEATURE - LOCAL SHAPE HISTOGRAM
The principal local feature extracted from the input is a local shape histogram. This is similar to previous approaches that partition space around a target point and count neighboring points falling inside these bins [Anguelov et al, 2005]. However, previous approaches orient the partitioned cube of space with respect to the principal plane around the target point. When classifying outdoor scenes, rotation with respect to the vertical usually contains a significant amount of information that can increase recognition accuracy compared to rotation invariant features. Thus, our approach includes the vertical vector when orienting the partitioned cube around a point. For every surface in our input set, we take the center point \( \bar{x} \), the normal vector \( \bar{n} \), and the up vector \( \bar{u} \) and define a local coordinate system as in

\[
[\bar{x} \ \bar{y} \ \bar{z} \ \bar{t}] = ((\bar{n} \times \bar{u}) \times \bar{u} \ \bar{n} \times \bar{u} 
\]

These local coordinates form the basis vectors for the partitioning space. Points contained in each bin of the cube are counted and stored as a normalized multi-dimensional histogram (i.e. if the dimensions of the partitioning cube is \( d \times d \times d \), then the histogram is a \( d^3 \) dimensional vector) as shown in Figure 2. This histogram can express the three dimensional shape around a given point while being invariant to rotation around the vertical axis. The differing shape results in sharply different histograms. Due to the need for obtaining all points within a defined area (in this case, the partitioning cube), we inserted all of the points into a KD-tree for high-performance distance-based lookups.

![Figure 2 Local coordinate system in range data and the local shape histogram (LSH).](image)

2.3 OBJECT FEATURES
To train object features for a training data set, we first manually segment the data into separate objects. Then, for each object \( q_i \), we compute the covariance matrix of the distribution of points composing it as in

\[
COV_i = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})(x_i - \bar{x})^T
\]

where \( x_i \) is the 3D Cartesian coordinate for each point in the object. The rows of the covariance matrix can be used to define a 3D box containing all of the points, providing a rough expression of the shape of the object. The label of the object is the label \( l \) of all the points comprising it. As such, the object feature for \( q_i \) is simply \([COV_i l]\). Fig. 3 shows sample manual object segmentation and boxes obtained from the basis vectors of the covariance matrix.

During recognition, input range images cannot be manually segmented into objects. As such, it is necessary to use an auto-segmentation technique to find objects in the input scene. As the range data has already been triangulated into polygons, as described in the section 2.2, it is trivial to search along
neighboring polygons to find all connected polygons

Figure 3 Object segmentations and shape features

\( x_j \) that make up an object \( o_i \) and calculate \( COV_i \).

Then, we match \( COV_i \) against all of the features in the object database using a Bhattacharyya distance function. A histogram of the \( k \) nearest entries is used to construct object class distribution \( P_o(l) \) by counting up the number of near neighbors for each class and normalizing to form a probability distribution.

2.4 LOCAL FEATURES

While it is important to create an extensive database encompassing a large sample of data to ensure high-accuracy recognition, the processing time of the algorithm is directly related to the size of this database. In order to take into account a large sample of data while maintaining a reasonable execution time, we employ a vector quantization technique to compress the sample features into representative clusters using the k-means++ algorithm to reduce the appearance of empty clusters [Arthur and Vassilvitskii, 2007]. Using a codebook has been shown to provide accurate results with relatively low execution time [Liebelt et al, 2008]. We ran clustering with respect to the local shape histograms (LSH) for each polygon. A histogram represents the shape feature of the polygon, so similar polygons will have similar histograms and should end up in the same cluster after k-means processing. After generating \( k \) clusters, it is important to determine the features associated with each cluster. Averaging the histograms of each element in the cluster can produce a reasonable representative shape feature for the cluster. To prevent one class from dominating the others by having a greater number of points among the training scenes, we ran the clustering on a class-by-class basis. For each class label \( l \), we produce \( k \) clusters of the histograms of all polygons \( x_i \) with label \( l \). The shape feature of the cluster becomes \( \bar{h} \), the average of all of the histograms of the elements in the cluster. The label for the class is the label of the elements in the cluster. As such, the resulting database has \( kL \) rows of \([\bar{h}]\), where \( L \) is the total number of classes.

Matching codebook entries are found for each input histogram using a brute force k-nearest-neighbors search for each input histogram. More sophisticated methods such as KD-trees are not appropriate due to the high dimensionality of the feature vector. \( n \) nearest codebook entries are found for each input histogram, and the number of matching entries for each class are counted up and normalized to produce class distribution \( P_l^c(l) \) for input histogram \( h_l \).

2.5 MARKOV RANDOM FIELDS FOR GLOBAL OPTIMIZATION

\( P_l^c(l) \) and \( P_o^c(l) \) are used to set up the potentials of a Markov network for final optimization. Markov networks can be modeled as in the following equation.

\[
P(0) = \frac{1}{Z} \prod_{i=1}^{k} \phi_i(l_i) \prod_{j \in i} \phi_{ij}(l_i, l_j)
\]

Here, \( \phi_i(l_i) \) is the tendency of node \( i \) to take on the label \( l_i \) and \( \phi_{ij}(l_i, l_j) \) is the tendency of two nodes with labels \( l_i \) and \( l_j \) respectively to be connected in the network. \( \phi_i(l_i) \) can be naturally expressed as a function \( p_i(l) \), the probability node \( i \) is an instance of class \( l \). This notion of probability is what led us to create probability distributions for each node in the previous sections rather than using a discriminative method similar to what has been employed in previous work [Anguelov et al, 2005]. Each polygon in the input scene is represented as a node in the Markov network with node potential as defined below:

\[
\phi_i(l_i) = w_p P_l^c(l_i) + w_o P_o^c(l_i)
\]

\( w_p \) and \( w_o \) are weights given to the shape class distribution and object class distribution respectively. Edge potentials are defined as

\[
\phi_{ij} = \begin{cases} 
\phi_L & \text{if } l_i = l_j \\
\phi_S & \text{if } l_i \neq l_j 
\end{cases}
\]

where \( \phi_L \) and \( \phi_S \) are user-defined constants such that \( \phi_L \geq \phi_S \).

We solve for a pseudo-optimal configuration for the Markov network using the alpha-expansion procedure introduced [Boykov et al, 2001] that uses an iterative minimum cut algorithm to guarantee a
For accurate classification of the Google Maps image, we took advantage of both the aerial photographs and logistical maps provided on Google Maps. Maps have the advantage of containing highly accurate data for roads and buildings, but they lack data about trees and narrow roads making it very difficult to align a range image using them alone. On the other hand, aerial images provide detailed information about the real state of the area, but it is not a trivial task to extract classes from just the photograph. By using both simultaneously, it is possible to cancel out the disadvantages to obtain relatively accurate classification results. Maps are trivial to classify as they have a 1-to-1 mapping from pixel color to class.

For aerial images, we extract image features as described below and match against a training set using K-Nearest Neighbors (K-NN) to obtain a class probability distribution for each pixel. This distribution is multiplied with the result from map classification to give a final class probability distribution.

### 3.1.1 IMAGE FEATURES

Classification of aerial images is done using two types of descriptors - texture and color. Texture features are effective at finding edges to identify buildings and roads, and color features are effective at finding trees and other natural objects. In this work, we used SURF [Bay et al, 2008] as a texture descriptor and Yxy color values as a color descriptor. As we calculate a class distribution for every pixel in the image rather than just at maximums, the fast execution speed offered by SURF proves valuable. As for color, since aerial images are taken in a variety of time and weather conditions, luminance varies greatly across images and it is important to choose a colorspace which separates luminance and hue. We took into consideration Yxy, HSV, L*a*b*, and L*u*v*, and in experimental tests, Yxy produced the best results.

### 3.1.2 CLASS DISTRIBUTION

The class distribution for each pixel is calculated by taking its 67-dimensional feature vector (SURF 64, Yxy 3) and applying K-NN against a training database. Vectors $f_a$ and $f_b$ are compared via a Manhattan distance function as described in equation:

$$d(a, b) = w_s \sum_{i=0}^{63} |f_a^i - f_b^i| + w_c \sum_{i=64}^{66} |f_a^i - f_b^i|$$

Here, $w_s$ and $w_c$ represent weights applied to the SURF and Yxy features respectively. Taking the K-nearest feature vectors in the training database along with their corresponding classes, we can calculate the class distribution of a pixel as the class histogram among the K vectors as in equation
\[
    p_{c}^{\text{KNN}} = \frac{N_{c}}{K}
\]

Here, \(N_{c}\) is the number of vectors of class \(c\) among the \(K\)-nearest vectors.

### 3.1.3 Final Classification by Map

Using only aerial image information, one ends up with many points that are difficult to classify; for example, the roof of a building and center of a road have very similar texture and color and cannot be readily distinguished. To get around this, we use the highly accurate information from a logistic map. However, these maps contain extraneous information such as border lines and text which cannot be mapped to a class. To eliminate this extraneous information, we apply a median filter several times to remove this noise. The RGB color value of the pixel is compared with a database mapping color to class distribution, and in cases where a perfectly matching color cannot be found in the database, we find the closest one and scale the probabilities with respect to distance.

Having \(p_{c}^{\text{MAP}}\) obtained from the map and \(p_{c}^{\text{KNN}}\) obtained from the aerial image, we can compute a probability distribution for a pixel as their normalized product, as shown in:

\[
    p_{c} = \frac{p_{c}^{\text{KNN}} p_{c}^{\text{MAP}}}{\sum_{c} p_{c}^{\text{KNN}} p_{c}^{\text{MAP}}}
\]

Figure 5 shows a color-coded image of a classified map.

![Figure 5 Classification result of google map image. Top-left: aerial photograph, Top-right: logical map, Bottom: Classification result.](image)

### 3.2 Google Maps Image and Range Data Registration

Having classified the Google Maps image and range data, the final step is finding the location and orientation of the range data with respect to the Google Maps image. To do this, we convert the 3D range data into a 2D template image and run a hierarchical template matching algorithm to find what portion of the classified map the template corresponds to.

Since Google Maps images are only 2D, it is necessary to drop the dimensionality of the range data from 3D to 2D for matching. Since the Google Maps images all have a top down view with known scale, it is sufficient to view the range data from a similar viewpoint, or in other words, to project the range data onto the ground. This is done by simply mapping the \(x\) and \(z\) coordinates of the range data onto a 2D plane. However, as range data is continuous as opposed to a discrete 2D image, a voting algorithm is used to combine the class distributions of multiple range data points that correspond to one 2D point.

Unfortunately, since laser range finders can only capture surface points, what you end up with is a very sparse 2D image which does not properly represent the volume of objects. For example, a building becomes simply a line in the template image as only its front-facing wall can be captured by the sensor. This makes it much more difficult to find a proper match for the template as useful information such as the buildings are mostly missing. To fill in the missing pixels, we use an interpolation step based employing a Gaussian random field.

Potentials vary by class to take into account its physical properties - for example, roads should extend forward indefinitely and trees should expand outward a certain amount. This is implemented by setting a Gaussian kernel of appropriate size. In addition, we want potentials to extend outward from the range sensor, since it is the area behind captured points that we want to fill in. To achieve this, we attempt to limit the effects of the Gaussian kernel by first creating an initial class distribution by searching and combining the points near the interpolation point in the direction of the sensor.

This initial distribution is then weighted with respect to the result of the Gaussian random field.

### 3.2.1 Hierarchical Template Matching

As simple matching of the template against the Google Maps image at every possible location and orientation is prohibitively expensive, we split the matching up into a series of hierarchical steps to find candidate regions for the template and then refine the search within these candidates. The final result is a probability map which represents the probability of the template matching a particular pixel location and the orientation that resulted in this probability.

The hierarchical template matching is done by taking a coarse-to-fine approach with respect to scale and rotation angle. We treat the template matching as a combination of 1 subproblems of varying scale. Each scale has a size \(S_{i}\) and angle width \(\Delta \theta_{i}\) where the images are shrunk to a factor of \(S_{i}\) and the template is rotated for all \(k\Delta \theta_{i}\) \((0 \leq k \leq 360/\Delta \theta_{i})\). The match probability \(P_{i}(x_{m}, y_{m})\) of the
template to the corresponding pixels of the image is defined as the sum of the Bhattacharyya distances of the class distributions across every pixel as shown in Equation:

$$p_t(x_m, y_m) = \sum_{x,y} \sum_{t} T^{i,k}_{t}(x_t, y_t) \cdot M^{i}_t(x_t + x_m, y_t + y_m)$$

Here, $T(x_t, y_t)$ is the probability that pixel $(x, y)$ in the template of scale $S_t$ and rotation $\alpha_t$ is class $C$, and $M^i_t(x_t + x_m, y_t + y_m)$ is similarly the class probability of the pixel in the Google Maps image. $(x_t + x_m, y_t + y_m)$ is the pixel in the Google Maps image that corresponds to the current template image pixel.

Having the matching probabilities $p_t(x_m, y_m)$ of each map pixel and their corresponding rotation angle $\alpha_t(x_m, y_m)$, we proceed with traditional template matching at areas of high probability down to a fixed threshold to obtain the exact matching angle $\theta(x_m, y_m)$. $p_t(x_m, y_m) = 0$ for pixels with probability below the threshold. In addition, we assume that the best match will occur at an angle near $\alpha_t$ and only take into consideration these nearby angles. Doing this, we obtain a probability map $p_t(x_m, y_m)$ for the Google Maps image and the corresponding optimal matching angle $\theta(x_m, y_m)$.

### 3.2.2 Registration

While we have calculated a probability map for the location and orientation of a given range image, it is not possible to simply contrive the highest probability point to be the actual result.

Architecture is made up of patterns, and it is often difficult even for a human to distinguish between two areas that look similar, and it is all the more difficult for an algorithm to distinguish between similar areas on the map using just class information. To solve this problem, we take the probability maps of several range images and a rough estimate of the actual distance between each range image to determine the actual locations of each image. The possible routes taken by the sensor can be modeled as a tree with nodes corresponding to pixel locations on the map, and a depth-first-search (DFS) is used to find the optimal route in this tree. The children $\text{Child}(x, y)$ for node $\text{N}(x, y)$ are all the nodes at a distance $D$ from the location as shown in the following Equation:

$$\text{Child}(x, y) = \{(x', y')|D - \varepsilon < ||(x, y) - (x', y')|| < D + \varepsilon\}$$

$\varepsilon$ is an error parameter for the distance between range images. The cost $\text{Cost}_n(x, y)$ at $\text{N}(x, y)$ of range image $n$ is defined in:

$$\text{Cost}_n(x, y) = -\log P_n(x, y)$$

$P_n(x, y)$ is the match probability of the n-th range image at position $(x, y)$. Where $P_n(x, y) = 0$, we insert an arbitrary fixed value to take into account possible error in the matching stage. Using this node cost, it is possible to define the cost of a given route passing through pixels $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$ as the sum of the node costs as in:

$$\text{Cost}((x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)) = \sum_n \text{Cost}_n(x_n, y_n)$$

We find the route with lowest cost using DFS. Branch-and-bound techniques are employed to be able to execute this DFS at reasonable speeds. Still, as the algorithm is exponential in the number of range images, we split the input range images into groups executed sequentially, with the result of a subsequent group extending from the result of the previous one.

### 4. Experimental Results

To test the effectiveness of the proposed method, we took nine outdoor scenes of Osaka University with a rotating SICK LMS-200 laser range finder mounted on a mobile robot. We selected these scenes in such a way that we could obtain a variety of samples for each of the classes under consideration. We then manually labeled all the points in all of the scenes with the correct class. In addition, we identified objects in the scene and assigned a unique label for each of these. We then selected representative data from four of these scenes so that we ended up with a database of 237177 points with their respective histograms and object features.

Histograms were calculated by a 80cm × 80cm × 80cm block partitioned into 8 × 8 × 8 bins. The histogram database was then clustered into 100 clusters for each class resulting in a final training database of 1K representative centroids.

Probability distributions for input nodes were calculated from the 100 nearest matching codebook entries. The training images contained 89 objects whose features were placed in the object database as is. Input object features were matched against their one closest database entry as the object database was still relatively small. $f_{10 n}, f_{11 n}, w_{0 n}$ were 0:1, 1:5, 1:0, 0:2 respectively. We then inputted the five remaining scenes into our recognition algorithm to infer the class labels for every point. These inferred labels were compared with manually assigned ground truth labels.
4.1 RANGE IMAGE CLASSIFICATION RESULTS

Scene is shown in Table 1 for MMM-classification and in Table 2 for classification without object features using only local shape histograms and alpha shown in Figure 6. Recall rates remained more or less consistent between the proposed and previous method. However, the recall rate for car was much improved by using object features. This was expected, as previous misrecognition of car usually involved doors being detected as building walls, but the object feature for car is able to match the entire vehicle, eliminating the building misrecognition in many cases. Step has a low recall rate, oftentimes misrecognized as car, but there are not enough samples in the test set to make a clear judgment as to why.

4.2 2D-3D REGISTRATION RESULTS

51 range images were taken around the Engineering Science building at Osaka University, Toyonaka campus. The GPS unit used was a Garmin eTrex Vista. In addition, actual positions and orientations were obtained by aligning and positioning the range images by hand. Weights for classification features were set to \( w_g = 1.0 \) and \( w_r = 0.6 \).

Figure 7 shows the results for position estimation. Blue represents the actual route of the sensor, green is the result of the proposed method, and red is the value obtained from GPS. The dashed-red line in the bottom left represents a location where the GPS could not return a value. The proposed method had an average relative error of 3.65m while GPS had a significantly larger error of 15.39m. In addition, absolute x and y error of the proposed method averaged (2.80, -3.20) with standard deviation (2.32, 2.38) while GPS had error (11.72, -9.20) with standard deviation (17.98, 41.78). These numbers show that the proposed method not only has high accuracy compared with GPS but very high precision which is important if one decides to employ an interactive post-processing step.

Figure 8 shows error values for orientation estimation of each image. The proposed method has average relative error of 24° with standard deviation 57.25 whereas GPS has an error of 16° with standard deviation 12.83. The high error of the proposed method can be attributed to six images which with approximately 10° or error. Using a purely class-based approach, a narrow road lined with buildings is essentially the same rotated even when 10°. Stripping out these six values, the relative error of the proposed method falls to 3° with standard deviation 3.34. An image showing the final alignment of the range data is shown in Figure 9.

Table 1 Total recognition rate for MMM classification

<table>
<thead>
<tr>
<th>Ground</th>
<th>Tree</th>
<th>Building</th>
<th>Step</th>
<th>Car</th>
<th>Percent</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>28/471</td>
<td>141</td>
<td>100</td>
<td>50</td>
<td>37</td>
<td>365</td>
<td>90.90%</td>
</tr>
<tr>
<td>Actual</td>
<td>299</td>
<td>90</td>
<td>1396</td>
<td>203</td>
<td>113</td>
<td>50.90%</td>
</tr>
<tr>
<td>Precision</td>
<td>99.60%</td>
<td>92.40%</td>
<td>84.01%</td>
<td>91.64%</td>
<td>90.51%</td>
<td>90.50%</td>
</tr>
</tbody>
</table>

Table 2 Total recognition rate for only local shape histograms with alpha expansion

<table>
<thead>
<tr>
<th>Ground</th>
<th>Tree</th>
<th>Building</th>
<th>Step</th>
<th>Car</th>
<th>Percent</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>28/471</td>
<td>141</td>
<td>100</td>
<td>50</td>
<td>37</td>
<td>365</td>
<td>90.90%</td>
</tr>
<tr>
<td>Actual</td>
<td>299</td>
<td>90</td>
<td>1396</td>
<td>203</td>
<td>113</td>
<td>50.90%</td>
</tr>
<tr>
<td>Precision</td>
<td>99.60%</td>
<td>92.40%</td>
<td>84.01%</td>
<td>91.64%</td>
<td>90.51%</td>
<td>90.50%</td>
</tr>
</tbody>
</table>

Figure 6 Classification result of a range image. Left: ground-truth, Right: Our result.

Figure 7 Position estimation result.

Figure 8 Orientation error of proposed method versus GPS.

Figure 9 The aligned 3D point clouds mapped onto a Google Maps Image.
ACKNOWLEDGEMENTS
This work was supported in part by the JST Precursory Research for Embryonic Science and Technology (PRESTO) program, the MIC Strategic Information and Communications R&D Promotion Programme (SCOPE), Japan.

8. REFERENCES