View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by ScholarlyCommons@Penn

University of Pennsylvania
ScholarlyCommons

Technical Reports (ESE)

Department of Electrical & Systems Engineering

2014

Towards a method for obstacle porosity classification

Sonia F. Roberts
University of Pennsylvania, soro@seas.upenn.edu

Follow this and additional works at: http://repository.upenn.edu/ese_reports

Part of the Electrical and Computer Engineering Commons, and the Systems Engineering Commons

Recommended Citation
Sonia F. Roberts, "Towards a method for obstacle porosity classification", *Technical Report (ESE)* . January 2014.

This paper is posted at ScholarlyCommons. http://repository.upenn.edu/ese_reports/13
For more information, please contact repository@pobox.upenn.edu.

Towards a method for obstacle porosity classification

Disciplines
Electrical and Computer Engineering | Engineering | Systems Engineering

This technical report is available at ScholarlyCommons: http://repository.upenn.edu/ese_reports/13

Towards a Method for Obstacle Porosity Classification

Sonia Roberts

April 2014

Introduction

A potential new application of the rough-terrain-ready six-legged robot RHex [1, 2] is for conducting long-term surveys of wind patterns in desert or desertifying environments. Of interest to aeolian research scientists is the frequency and *porosity* of obstacles in the environment, which affects the ability of wind gusts to carry small soil particles over long distances, affecting the fertility of the soil. Placement, frequency and porosity of obstacles such as vegetation affect downstream wind patterns [3, 4, 5], with small changes in vegetation resulting in large, nonlinear changes in wind behavior [6, 7].

The task is to use LIDAR data, possibly augmented with a color camera, to determine the *environmental roughness* of a large area through which the robot is permitted to perform an arbitrary walk. In this context, environmental roughness is a metric for the amount of disruption to wind gusts carrying small, fertile soil particles a given area produces, which is relevant to research on long-term weather patterns in general and the serious issue of desertification in particular [3, 4, 5]. Environmental roughness is determined as a function of the frequency, size, and *porosity* of obstacles in the environment, where porosity is a metric for how disruptive to a wind gust a particular object is, with a porous obstacle being less disruptive than a solid one.

Usually, aeolian environmental surveys are conducted by humans who visually examine the landscape, looking for obstacles and noting their size, placement relative to each other, and porosity. For a robot application, detecting the percentage of area encountered that is open versus obstructed and gauging the porosity of the obstacles using LIDAR should be sufficient for a robot conducting a long-term survey to gain an estimate of the roughness of the area.

The learning problem can be broken into three parts. During a walk through the environment, the robot needs to identify obstacles that may warrant further investigation for porosity classification purposes. Upon identifying and orienting towards one such obstacle, it should be investigated to collect additional data. Finally, the obstacle must be classified as having some particular porosity.

The first part of the task requires only the identification of a group of pixels as an "open area", "porous obstacle", or "solid obstacle", and must be computed on-line. However, it is not so important if the response is biased so long as it is biased towards porous obstacles, which require further investigation. The second part of the task, the investigation of a potential object of interest, could require circumnavigation of the obstacle (creating a loop closure problem) or the robot simply orienting towards the obstacle and wiggling back and forth or up and down to acquire more information. The third part of the task requires that the robot compare the data taken during the investigation of the obstacle with data taken from obstacles of known porosities. This part of the task could be performed later, after the data is downloaded from the robot, and therefore does not carry the same computational speed constraint. However, obstacles of greater and lesser porosities must be distinguished between in a much more fine-grained manner than in the first part of the problem. Thus, both bias and variance are more important to minimize, but the costs of increasing model complexity in computational time are less.

I have already attempted to address the first part of this task, the rough identification of obstacles of interest, in the final project for the Learning in Robotics course, in which I attempted to detect solid and porous obstacles using only depth information quickly enough to be useful to a robot as it walks around. For

this task, I used a LIDAR and color video dataset collected by a labmate for a different set of experiments testing an obstacle-avoidance controller in an open-forested environment with trees, bushes, and branches on the ground. I stacked all the LIDAR scans from each timestep and used a "moving window of interest" around each pixel, using information from previous timesteps as well as information about the pixels to the left and right, to classify the individual pixels. I took the approach of considering the pixels in the "window" as a series of independent random samples of a probability variable X. I constructed three template distributions from a training sample of the LIDAR data using video for ground truth: Solid obstacle, porous obstacle, and open area. I then classified each pixel based on its metric distance from each of the three template distributions.

I compared performance of several classical methods for distribution distance measurement to use as my metric, including the Kolmogorov distance, Kullback-Leibler divergence, a simple L_1 and L_2 norm, and an interesting new method for determining whether a collection of distributions is similar which uses an algorithm called the bounded L_∞ closeness [8], so called because it relies on the assumption that the distributions being tested all have finite support. However, the most reliable metric, and the only one that could be computed quickly enough to be used online, was a very simple rank-order statistic: The percent of LIDAR readings in the window that were below a threshold. Since the open area and solid obstacle distributions ended up looking roughly Gaussian and the porous obstacle distribution ended up looking qualitatively like a bimodal Gaussian distribution (see Figure 1), this percentage actually varied greatly between the three template distributions. Furthermore, the absolute difference of the percentages p of two samples $d(a, b) = |p_a - p_b|$ easily satisfies the criteria for a metric[1] by the same argument as the metric induced by the one-dimensional L_1 norm.

The second part of the task is only difficult as a loop closure problem. Assuming its treatment as a loop closure problem, I anticipate that the tractability will be greatly increased by the incorporating odometry, GPS, color video, and accelerometer information. Therefore, I will not attempt a pure LIDAR solution, and will not discuss one here.

The third part of the task, the fine-grained porosity assessment of an obstacle of interest, will be considered here. Whereas speed of computation was the highest priority in the rough identification of obstacles of interest, resulting in the selection of a very quick-to-compute rank-order statistic, this fine-grained assessment of an obstacle does not necessarily need to be performed online. I will again take a distribution comparison approach and discuss different metrics for evaluating the distances between distributions. Results from the porous obstacle identification task I performed for the Learning in Robotics course will be discussed as they apply to the porosity assessment problem.

Methods explored previously

To address the first part of the task, the rough identification of solid obstacles, porous obstacles, and open areas, I used the LIDAR data and accompanying color video taken by my labmate Deniz Ilhan for one of his obstacle avoidance experiments using a RHex at Ridley Creek State Park. RHex is a six-legged robot that walks at approximately human speeds (5mph) using an alternating tripod gait and successfully climbs over most rough terrain that is encountered in natural environments relevant to desertification. In these experiments, RHex carried a Hokuyo LIDAR scanner at a height of 12 inches off the ground. The LIDAR scanner takes a single horizontal sweep per data capture instance, with a capture rate of about 10 scans per second. For this experiment, the sensitivity was set to a range of 1-5m, which means that any obstacles further than 5m away were not detected; the scanning angle was 270 degrees and 681 pixels were captured per scan.

I stacked the first 500 LIDAR scans so that they appeared as a black-and-white "image" and watched the video from the first few seconds carefully to see what the robot was looking at. Then, I used Matlab's roipoly function to create three training sets of pixels in the three desired distributions: "open area",

[1] It is non-negative, equals 0 only when the two percentages are the same, is symmetric, and satisfies the triangle inequality $d(a, b) + d(b, c) = |p_a - p_b| + |p_b - p_c| \geq |p_a - p_b + p_b - p_c| = |p_a - p_c| = d(a, c)$.

"porous obstacle", and "solid obstacle". Histograms of the three training sets are included in Figure 2, which demonstrate that each of the three distributions do have visually distinct shapes.

I treated the LIDAR scan as a very large image and classified each pixel in this "image" as coming from an open area, a porous obstacle, and a solid obstacle. Each pixel was classified using a moving window of 40x40 pixels. Note that this takes in historical information about a pixel (as time is in the "y axis" of the "image") as well as information about what is to the left and right of the pixel. Also, since the image was not stabilized by accelerometer data, some noise was introduced when the robot turned to the left or right. The 1600 pixels from this 40x40 window were taken to be a sample drawn from a probability distribution which was then compared with the template distributions for open area pixels, porous obstacle pixels, and solid obstacle pixels.

Classification was determined by the distance[2] of the probability distribution calculated from the histogram of the 1600-pixel sample to each of the three template distributions, with the minimum distance determining the classification. I compared classification performance with six different methods for distance measurement, most of which I implemented and one of which was available through an existing Matlab function. The Kolmogorov distance [9, 10, 11], L_p norms, and Kullback-Leibler divergence [12] are all standard, well-known distance measuring methods for distributions that have been around for decades. The bounded $L-\infty$ closeness test was developed in a recent paper [8] and I chose to implement it for this project because it seemed like an interesting general-case test for distances between distributions. Notice that this is *not* an infinity norm. Rather, it gets its name from its assumption that all distributions being compared have finite support (or equivalently, bounded L_∞ norms). I also implemented a very simple rank-order statistic method just looking at the percent of pixels in the sample window for which there were no obstacles detected in the 5m sensitivity radius.

The rank-order statistic method performed the best and did so with the fastest run-time. The traditional methods for distance measurement between distributions (Kolmogorov distance, L_p norms, Kullback-Leibler divergence) did almost as well as the rank-order statistic but took substantially longer to compute. Finally, the bounded L_∞ norm closeness test performed horribly, being unable to distinguish between porous obstacles and open areas at relevant window sizes. In the following I discuss each method and its failure or success individually.

Not all of these six "distance measurement methods" are truly metrics, satisfying the mathematical definition.[3] The Kolmogorov distance and the metrics induced by the L_p norms are true metics. The rank-order statistic is a pseudo-metric; that is, the distance between two distributions by the metric induced by this statistic may be 0 even if the distributions are the same. The Kullback-Leibler divergence does not satisfy the symmetry or triangle inequality properties and therefore is not even close to a true metric, but it is a commonly used method for testing goodness-of-fit between sample and model distributions. Used as it is here with the template distributions always in the second position, it does provide a constant classification scheme. The bounded L_∞ closeness test is also not a true metric, and is not guaranteed to satisfy any of the metric properties because its result depends on the random samples drawn from the distributions it is comparing.

In the following, I will discuss each distance measurement method individually. I will then compare their performances and present some results.

Kolmogorov distance

I used the Kolmogorov-Smirnov statistic as a distance metric, which is defined as the maximum difference between the sample cumulative distribution and its purported template cumulative distribution:

$$Kol(P,Q) = \max_i |P(i) - Q(i)|$$

Notice that this is basically an infinity norm if the two distributions being compared are discretized into n bins (where n is the sample size), and the weight on each bin becomes its value in an n-dimensional

[2] Not all of the methods for calculating distance used were true metrics, but I will use the term "distance" generally.

[3] Nonnegative; symmetric; equaling 0 when the two distributions are the same; triangle inequality.

space. Thus, for example, a uniform probability distribution taking weight between 0 and 1 with 4 bins would be a discretized into a histogram with weight 0.25 at each of its four bins, and the associated vector would be $(0.25, 0.25, 0.25, 0.25)$. I wrote a distance function for the Kolmogorov distance, which is included in the appendix. This metric did fairly well; it had trouble distinguishing between porous obstacles and solid obstacles, but it is possible that more careful window size adjustment or more training data would solve that problem. The worst failures occurred during servoing maneuvers.

L_1 and L_2 norms

The L_1 and L_2 norms (and any other L_P norm) take the approach of considering a discrete probability distribution with support in set R as a finite-dimensional vector with values of each index i corresponding to the probability associated with R_i. The distance is then just the distance induced by the norm in any Euclidean space, where the L_p norm is defined as:

$$L_p(x) = \left(\sum_i x_i^p \right)^{1/p}$$

I wrote distance metric functions for L_1 and L_2 which are included in the appendix. These methods gave almost exactly the same results as the Kolmogorov distance.

Kullback-Leibler divergence

The Kullback-Leibler divergence [12] of two discrete probability distributions, P and Q, is defined as follows:

$$\mathrm{KL}(P, Q) = \sum_i \ln \left(\frac{P(i)}{Q(i)} \right) P(i)$$

I downloaded an existing example of this code from the Mathworks website written by Joachim Selke; the function is called kldiv. I have included in the appendix the code that I wrote to call this function to calculate the distances between each point in the LIDAR scan and each of the three template distributions. This method was more heavily biased towards classifying both open areas and solid obstacles as porous obstacles.

Bounded L_∞ closeness test

The bounded L_∞ closeness test is designed to compute similarities between collections of distributions [8]. It gets its name from the assumption that both distributions have finite support – that is, the L_∞ norms of the support of each are bounded. Furthermore, it assumes discrete or discretized probability distributions. The high-level view of the test is that it measures the similarity between two distributions, p and q, by drawing two samples from each distribution, $F_p^1, F_p^2, F_q^1, F_q^2$ of a size determined by the size of the support R, known bounds on the L_∞ norms of the distributions b_p and b_q, and a confidence threshold ϵ.

A rough sense of the shape of each distribution is gained by considering the collisions in F_p^1 and F_q^1, where a *collision* is the occurrence of two sample values being the same. Collisions will occur more frequently in the heavy parts of the distribution, giving a rough sense of the shape within a distribution. Then, the collisions between samples F_q^2 and F_p^2 from the two distributions are considered. This provides a sense of the overlap of the two distributions. Distance is then determined by the number of collisions within a distribution versus across the two distributions. The algorithm is written out in Figure 3.

For this project, I implemented this test by sampling uniformly from each template distribution to get F_q^1 and F_q^2 and randomly dividing the window for each pixel into two samples F_p^1 and F_p^2. Unfortunately, given the bounds on the support and limited by an appropriate size for a moving window, the highest confidence I could guarantee for this test (determined by ϵ in the algorithm) was around 75%. The test itself has an additional uncertainty associated with it (Theorem 5.3 in the paper): If the L_1 norm of the difference between

the probability distributions p and q being compared is greater than the confidence parameter ϵ, $||p-q||_1 > \epsilon$, the test is only guaranteed to reject with probability greater than 2/3; likewise, if $||p-q||_1 \leq \epsilon/(2|R|^{1/2})$, where R is range of the support, it is only guaranteed to accept with probability greater than 2/3. This means that in the worst-case scenario, there is only a 50% chance that the test will provide the correct answer, given the sample size available.

Rank-order statistic

The rank-order statistic was based on the observation that in each probability distribution for a classification type (open area, porous obstacle, solid obstacle), there is a large difference in the weight of the distribution that is focused on the smallest bin, which corresponded to no reflective surface detected within 5 meters for that pixel. To calculate the "distance" in this case for each pixel in the LIDAR image, I simply computed the percent of pixels in the associated moving window below the threshold value (number of bins)$^{-1}$ that would have fallen into the first bin of a discretized probability distribution. The number of bins I used for this project was 20.

Classification into open area, porous obstacle, and solid obstacle categories was determined by thresholding on the percent of pixels: if a sample window had more than 80% low-valued pixels, it was determined to be an open area; between 30 and 80%, a porous obstacle; and fewer than 20% made it a solid obstacle. The thresholds were determined by visually examining the cumulative distribution functions for each of the three distributions testing a few options to see what parameters gave the best performance.

Comparison of metrics

The Kolmogorov distance and L_p norms all provided similar classifications, with the KL divergence and the rank-order statistic being more biased towards classifying open areas as porous obstacles. The bounded L_∞ norm closeness test performed terribly, with the classified LIDAR scan looking very noisy. See Figure 4 for a close-up example of the L_2 norm, as an illustrative example of the human-readable output of the classifier, and Figure 6 for a comparison of the whole LIDAR scan sequences from each of the six methods.

In general, the Kolmogorov distance, L_p norms, and Kullback-Leibler divergence *did not* tend to misclassify open areas as obstacles, nor did classify obstacles as open areas, which is a much worse failure. The exception was during abrupt servoing maneuvers, when these methods often failed to classify an object as solid during the turn. Porous obstacles and solid obstacles were most easily confused when the robot was servoing back and forth, which can be seen in the L_2 example in Figure 4. I suspect that this is because I considered the LIDAR scan as a single image and did not take the robot's orientation in world coordinates into account, meaning that abrupt turns leading to drastically different stimuli for the same pixels from timestep to timestep could easily confuse the classifier. I anticipate that either updating the moving window to be wider than it is tall (decreasing the amount of backwards and forwards time that is taken into account for pixel classification), providing more training examples that include many sharp turns, and taking into account the orientation of the robot in world coordinates to provide a stabilized view of the world from the point of view of the LIDAR might all improve performance.

The rank-order statistic did not fare quite as well, confusing porous and solid obstacles more frequently and failing sometimes to recognize thin obstacles as obstacles at all. See Figure 5 for an example of the failure modes of the rank-order statistic. However, these failures were most common when the robot was servoing and when it was walking straight it performed comparably to the other three methods previously discussed; see Figure 6. The thresholds could be adjusted to bias the classifier more towards detecting obstacles, which might introduce a systemic bias but would be more applicable for the proposed task. Therefore, I think this method is actually the best suited to the first part of the obstacle porosity detection problem, as the bias issue is less important and the computation time is very low.

The bounded L_∞ norm closeness test was unable to correctly classify the majority of pixels in the LIDAR scan, as can be seen in Figure 6. After testing that the closeness test correctly gave a low distance between each template distribution and itself, and a higher distance between the template open area distribution and the solid obstacle distribution than either to the porous obstacle distribution, I suspect that this is

because the sample window size is not large enough to ensure a desirable accuracy level. Because the test is probabilistic, and the first step is to divide the sample window into two samples to compare within the samples and between the sample window and the template distributions, there is no guarantee that the test will give a distance of 0 even when the distributions being compared are the same distributions; however the probability of this error is minimized as the sample size increases. It is possible that this method might actually produce very good results if the distributions being compared were not made up of such small samples. Data from the further investigation of a potential object of interest, for example, might enable better classification performance.

Proposed methods

For the first part of the obstacle porosity detection problem, the rough identification of obstacles of interest for further investigation, the metric distance nearest-neighbor classification scheme with the rank-order statistic provides the requisite very quick, biased classification. Furthermore, the rank-order statistic performed well in the preliminary implementation for the Learning in Robotics class, with comparable results to established distribution distance measurement methods even though its method was much simpler.

The second part of the problem, the further investigation of a potential obstacle of interest, will not be considered here. Recall that potential implementations would be circumnavigation of the object to gather information from all directions; servoing the robot left and right or up and down to gather more LIDAR scans from the same vantage point; or some combination of these two approaches. For the purposes of discussing methods for the third part of the problem, I will assume that the amount of data required for a given method is not a limiting factor.

The third part of the problem, the fine-grained classification of obstacles into different porosity categories, could be approached from a regression standpoint or a classification standpoint. That is, given example template distributions for obstacles of known porosities, the two main choices would be to (1) regress a line between the least porous obstacle distribution and the most porous obstacle, using the weights of each bin in the histogram describing each of the template distributions as the definition of a point in a vector space; or (2) compare sample distributions directly with the template distributions, using a nearest-neighbor scheme to classify each sample distribution as coming from the template distribution from which it has the smallest distance, as in the rough identification task.

Between these two choices, I suggest that the latter is the preferred approach. While porosity should really be a continuous variable, aeolian research scientists use a discrete classification scheme with only a few options during the human surveying procedures that we are attemping to automate. Thus, available to us will likely be only a few (3-5) porosities of example distributions from obstacles of known porosities, which is not enough to regress a reasonable line without significant errors from variance. The input space may also have to be quite large to accommodate the number of bins required to accurately capture the shapes of each of the template distributions or to lift the binned template distributions to a high-enough dimension that the regression may be linear. As discussed in chapters 2 and 7 of the textbook, clusters which are dense in low-dimensional spaces may not be as dense in high-dimensional spaces, making clustering and regression problems more difficult to find convergent solutions to; this is one of the motivations for limiting model complexity. Also, however theoretically nice it might be to have a continuous output variable, since the aeolian research scientists would ordinarily be collecting ordinal data it makes little sense to provide a more detailed response than they need if there is additional cost.

Having chosen the second approach, a few options remain to model the training data and classify new data. First, I could use maximum likelihood estimation to create a mixed Gaussian model of each the clusters of example distributions in the input space. Alternatively, I could use the approach that I used in the preliminary analysis, which is to simply lump all the distributions from any desired classification together into one template distribution for each possible output, and then consider new sample distributions to be sampled from one of those template distributions. For the initial problem, this second approach is efficient because it limits the number of comparisons necessary, which was extremely important for limiting the computation time. However, if the classification can occur later and off-line, then I propose that the former

approach is preferred, using either of the L_p metrics or the Kolmogorov distance to measure the distances between template distributions and create the mixed Gaussian models used in the expectation maximization procedure, as these all performed well and with lower bias than the Kullback-Leibler divergence or the rank-order statistic.

Now the motivation for the choice of metric moves towards minimizing computational time and complexity. Since in this implementation all are L_p norms (with the Kolmogorov distance being an infinity norm), the most computationally simple metric is the L_1 norm. Therefore, I propose to fit template distributions for obstacles of three to five known porosities using a mixed Gaussian model found by maximum likelihood estimation. Then, each new sample distribution will be classified by finding the Gaussian it is closest to in the input space, and assigning it the classification of that Gaussian.

For comparison, I propose giving the bounded L_∞ norm closeness test another try, with template distributions for each of the obstacle porosities determined by lumping together the example distributions for obstacles of that porosity. Since the test assumes access to two samples from each distribution being compared, it may be robust to the potential noise accrued by lumping together potentially very different example distributions. Furthermore, with sufficient data the limitations on classifier capability should be much reduced. However, since this procedure is very new in the literature, it is advisable to take the conservative approach and implement it only as a comparison to the maximum likelihood estimation method using the L_2 norm proposed in the previous paragraph.

Obviously, a new dataset will need to be collected which includes realistic robot "investigations" of obstacles of interest. I have already begun collecting this dataset using bushes of different porosities around Penn Park. I will explore other areas in Philadelphia this summer and expand to obstacles in different environments, in particular dune shrubs, at the end of the summer. Obstacle porosity classification will of course also need to be verified by expert collaborators in the aeolian research sciences.

A Kullback-Leibler code

```
function dists = calcKLdists(data, compDist, windowSize, nBins)

dists = zeros(size(data));

eps = 1/1000*ones(size(compDist));

X = 0:1/nBins:1;
X = X';

compDist = compDist + eps;
compDist = compDist/sum(compDist);

for i = windowSize:size(data,1)-windowSize
    for j = windowSize:size(data,2)-windowSize

        window = data(i+1:i+windowSize, j+1:j+windowSize);

        window = reshape(window,windowSize*windowSize,1);

        binned = binFreq(window, nBins);
        probDist = calcProbDist(binned, nBins);

        probDist = probDist(:,2);
```

```
        probDist = probDist + eps;

        probDist = probDist/sum(probDist);

        dists(i,j) = kldiv(X, probDist, compDist);

    end
end

end
```

B Kolmogorov code

```
function dists = calcKolDists(data, compCdf, windowSize, nBins)

dists = zeros(size(data));

for i = windowSize:size(data,1)-windowSize
    for j = windowSize:size(data,2)-windowSize

        window = data(i+1:i+windowSize, j+1:j+windowSize);

        window = reshape(window,windowSize*windowSize,1);

        binned = binFreq(window, nBins);
        probDist = calcProbDist(binned, nBins);

        probDist = probDist(:,2);

        cdf = pdfToCdf(probDist);

        comp = abs(cdf-compCdf);

        dists(i,j) = max(comp);

    end
end

end
```

C L_1 and L_2 norm code

```
function dists = calcL1Dists(data, compCdf, windowSize, nBins)

dists = zeros(size(data));

for i = windowSize:size(data,1)-windowSize
    for j = windowSize:size(data,2)-windowSize

        window = data(i+1:i+windowSize, j+1:j+windowSize);
```

```
        window = reshape(window,windowSize*windowSize,1);

        binned = binFreq(window, nBins);
        probDist = calcProbDist(binned, nBins);

        probDist = probDist(:,2);

        cdf = pdfToCdf(probDist);

        comp = cdf-compCdf;

        dists(i,j) = norm(comp,1);

    end
end

end

function dists = calcL2Dists(data, compCdf, windowSize, nBins)

dists = zeros(size(data));

for i = windowSize:size(data,1)-windowSize
    for j = windowSize:size(data,2)-windowSize

        window = data(i+1:i+windowSize, j+1:j+windowSize);

        window = reshape(window,windowSize*windowSize,1);

        binned = binFreq(window, nBins);
        probDist = calcProbDist(binned, nBins);

        probDist = probDist(:,2);

        cdf = pdfToCdf(probDist);

        comp = cdf-compCdf;

        dists(i,j) = norm(comp,2);

    end
end

end
```

D Closeness test code

```
function dists = calcblinfDists(data, compBinned, windowSize, nBins)

% distOp(:,:,1) will contain the distance information
```

```
dists = zeros(size(data,1),size(data,2));

% "Uniformly sample" Fq1 and Fq2 from the template
sampleTemplate = [zeros(length(compBinned)-windowSize/2,1);ones(windowSize/2,1)];
sampleTemplate = sampleTemplate(randperm(length(sampleTemplate)));
Fq1 = compBinned(logical(sampleTemplate));

sampleTemplate = sampleTemplate(randperm(length(sampleTemplate)));
Fq2 = compBinned(logical(sampleTemplate));

figure
subplot(1,2,1)
hist(Fq1)
subplot(1,2,2)
hist(Fq2)

for i = 1+windowSize:size(data,1)-windowSize

    for j = 1+windowSize:size(data,2)-windowSize

        window = data(i:i+windowSize-1,j:j+windowSize-1);
        dists(i,j) = blinfclose(window, nBins, Fq1, Fq2);

    end

end

end
```

E Rank-order statistic code

```
function dists = calcNaiveDists(data, windowSize, nBins)

dists = zeros(size(data));

for i = windowSize:size(data,1)-windowSize
    for j = windowSize:size(data,2)-windowSize

        window = data(i+1:i+windowSize, j+1:j+windowSize);

        window = reshape(window,windowSize*windowSize,1);

        window = sort(window);

        thresh = 1/nBins;

        op = find(window>thresh,1);

        if isempty(op)
            op = windowSize*windowSize;
        end
```

```
        op  = op/(windowSize*windowSize);

        dists(i,j) = op/(windowSize*windowSize);

    end
end

end
```

References

[1] G. C. Haynes, J. Pusey, R. Knopf, A. M. Johnson, and D. E. Koditschek, "Laboratory on legs: an architecture for adjustable morphology with legged robots," in *SPIE Defense, Security, and Sensing*, pp. 83870W–83870W, International Society for Optics and Photonics, 2012.

[2] U. Saranli, M. Buehler, and D. E. Koditschek, "Rhex: A simple and highly mobile hexapod robot," *The International Journal of Robotics Research*, vol. 20, no. 7, pp. 616–631, 2001.

[3] S. Wolfe and W. Nickling, "Shear stress partitioning in sparsely vegetated desert canopies," *Earth Surface Processes and Landforms*, vol. 21, no. 7, pp. 607–619, 1996.

[4] M. Raupach, "Drag and drag partition on rough surfaces," *Boundary-Layer Meteorology*, vol. 60, no. 4, pp. 375–395, 1992.

[5] G. S. Okin, "A new model of wind erosion in the presence of vegetation," *Journal of Geophysical Research: Earth Surface (2003–2012)*, vol. 113, no. F2, 2008.

[6] B. Marticorena, M. Kardous, G. Bergametti, Y. Callot, P. Chazette, H. Khatteli, S. Le Hégarat-Mascle, M. Maillé, J.-L. Rajot, D. Vidal-Madjar, *et al.*, "Surface and aerodynamic roughness in arid and semiarid areas and their relation to radar backscatter coefficient," *Journal of geophysical research*, vol. 111, no. F3, p. F03017, 2006.

[7] J. A. Gillies, W. G. Nickling, and J. King, "Shear stress partitioning in large patches of roughness in the atmospheric inertial sublayer," *Boundary-layer meteorology*, vol. 122, no. 2, pp. 367–396, 2007.

[8] R. Levi, D. Ron, and R. Rubinfeld, "Testing properties of collections of distributions," *Theory OF Computing*, vol. 9, pp. 295–347, 2013.

[9] A. N. Kolmogorov, "Sulla determinazione empirica di una legge di distribuzione," *Giornale dell'Istituto Italiano degli Attuari*, vol. 4, no. 1, pp. 83–91, 1933.

[10] F. J. Massey Jr, "The kolmogorov-smirnov test for goodness of fit," *Journal of the American statistical Association*, vol. 46, no. 253, pp. 68–78, 1951.

[11] H. W. Lilliefors, "On the kolmogorov-smirnov test for normality with mean and variance unknown," *Journal of the American Statistical Association*, vol. 62, no. 318, pp. 399–402, 1967.

[12] S. Kullback and R. A. Leibler, "On information and sufficiency," *The Annals of Mathematical Statistics*, pp. 79–86, 1951.

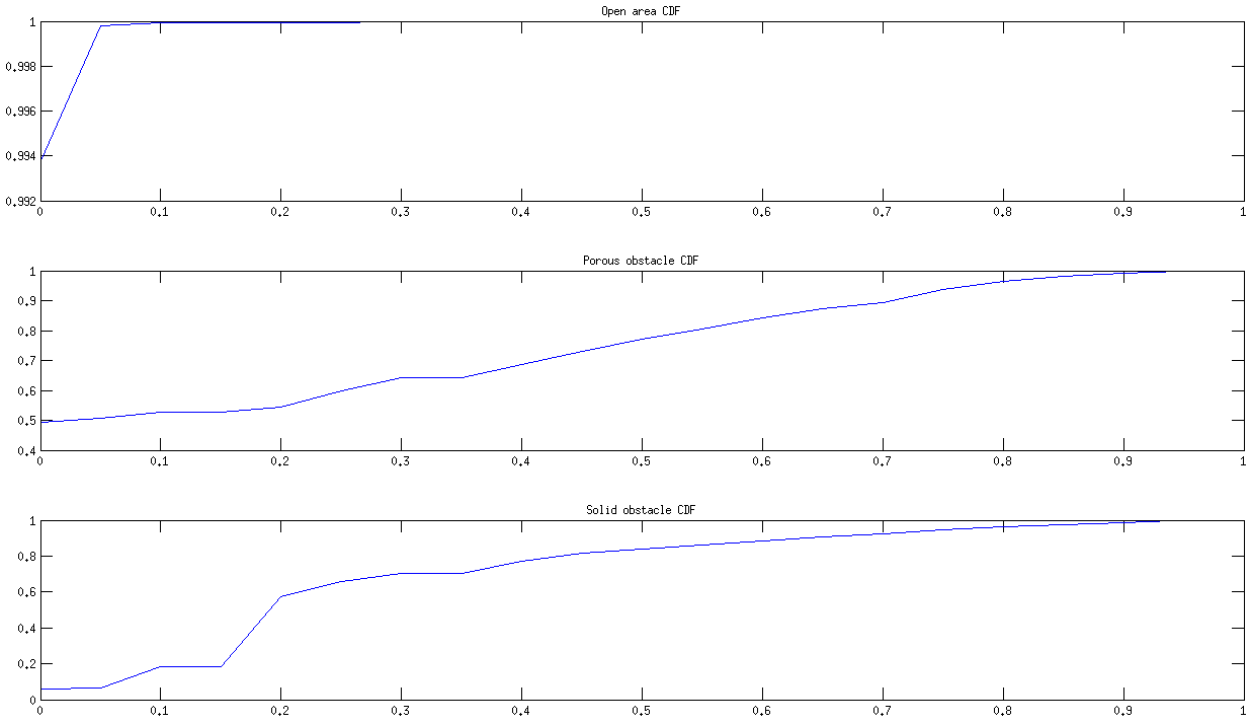

Figure 1: The cumulative distribution functions for the open area, porous obstacle, and solid obstacle template distributions that I used to determine the appropriate thresholding values for the rank-order statistic approach.

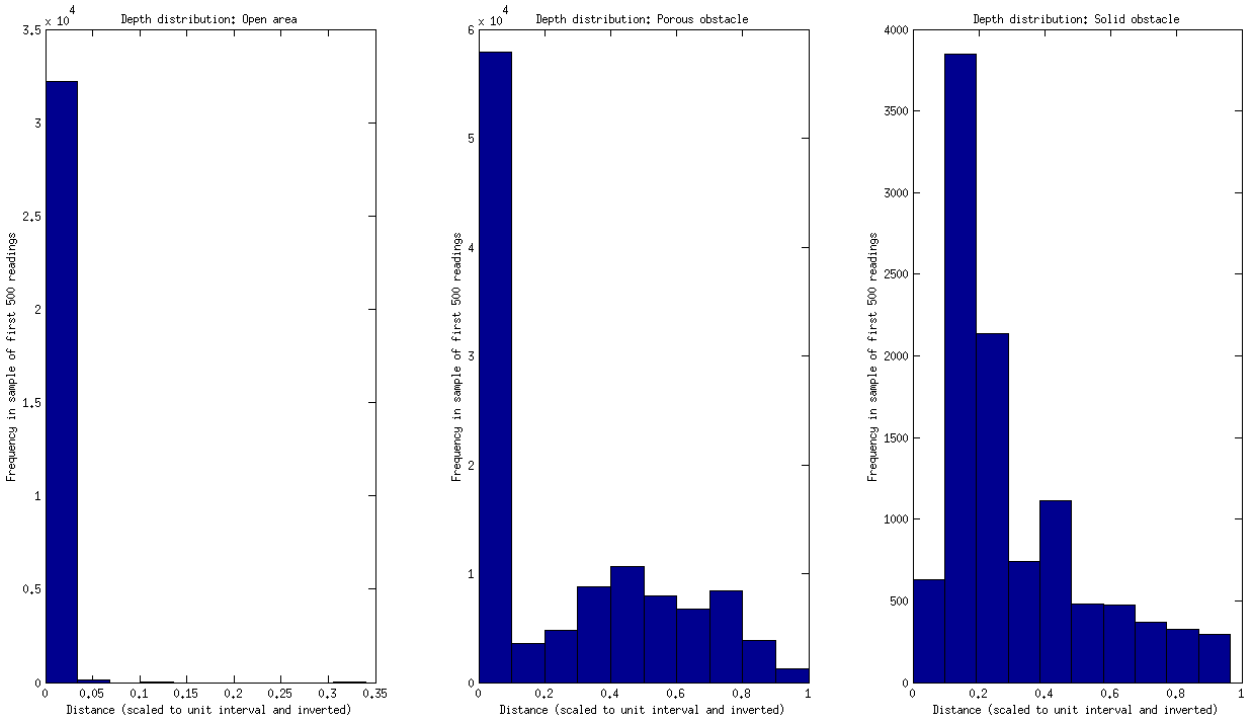

Figure 2: Histograms of the raw distance information from the LIDAR for open areas, porous obstacles, and solid obstacles. Scaled versions of this data served as the template probability distributions for each of the three classification categories. Notice that the distributions are all qualitatively different: The open area distribution has almost all of its weight at 0, the solid obstacle looks like a skewed Gaussian with the majority of its weight around 0.2 (or possibly like a mixed Gaussian model with one Gaussian centered at 0.2 and one centered at 0.4), and the porous obstacle distribution looks like a mixed Gaussian model with the one centered at 0 with very low variance and either one or two Gaussians centered at higher values.

Bounded-ℓ_∞-Closeness-Test (sampling access to p, sampling access to q, b_p, b_q, $|R|$, $\varepsilon > 0$)

1. Take samples F_p^1 and F_p^2 from p, each of size t, where $t = O\left(|R| \cdot b_p^{1/2}/\varepsilon^2 + |R|^2 \cdot b_q \cdot b_p/\varepsilon^4\right)$
2. Take samples F_q^2 and F_q^2 from q, each of size t
3. Let $r_p = \mathrm{coll}(F_p^1)$ be the number of *collisions* in F_p^1
4. Let $r_q = \mathrm{coll}(F_q^1)$ be the number of *collisions* in F_q^1
5. Let $s_{p,q} = \sum_{j \in R}(\mathrm{occ}(j, F_p^2) \cdot \mathrm{occ}(j, F_q^2))$ be the number of collisions between F_p^2 and F_q^2
6. If $r_q > (7/4)\binom{t}{2} b_p$ then REJECT
7. Define $\delta \stackrel{\text{def}}{=} \varepsilon/|R|^{1/2}$, $r \stackrel{\text{def}}{=} \frac{2t}{t-1}(r_p + r_q)$ and $s \stackrel{\text{def}}{=} 2s_{p,q}$
8. If $r - s > t^2\delta^2/2$ then REJECT, otherwise ACCEPT

Figure 3: The algorithm for the bounded L-infinity closeness test. Here, p and q are the probability distributions being compared, with b_p and b_q their L_∞ norms and $|R|$ the range of the support. The parameter ϵ determines the desired certainty level of the test. The number of collisions in a sample is a measure for where the weight of the distribution is. For $j \in R$, let $\mathrm{occ}(j, R)$ be the number of times that j occurs in the sample, and let the collision count of the sample be $\mathrm{coll}(F) = \sum_{j \in R} \binom{\mathrm{occ}(j,F)}{2}$.

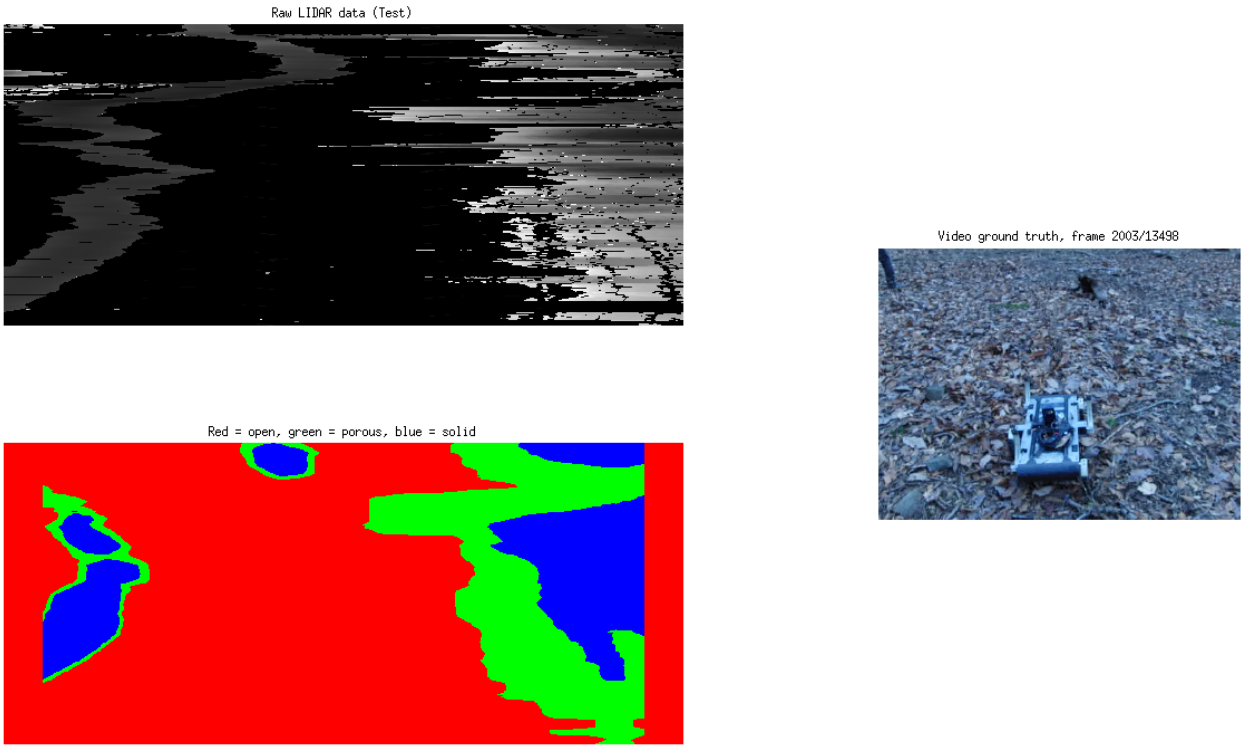

Figure 4: This figure illustrates the performance of the L2 norm, which was very similar to the performances of the Kolmogorov distance, both L_1 and L_2 norms, and the Kullback-Leibler divergence. In the upper left image is the raw LIDAR scans, each of which is a row of pixels, with the current scan at the top of the screen and the scans for future timesteps stacked below. The video frame associated with this scan, taken by a human following the robot, is on the right. The robot is negotiating a tree to the left (the trunk of which can be seen) and a large branch or small bush to the right, which is more difficult to pick out in this image. There is also a stump which is too short to be seen by the laser scanner. Most of the pixels in the tree trunk are correctly identified as solid obstacles, and the branch is identified as having some solid and some porous components. However, these methods tended to lose track of solid obstacles during abrupt servoing maneuvers.

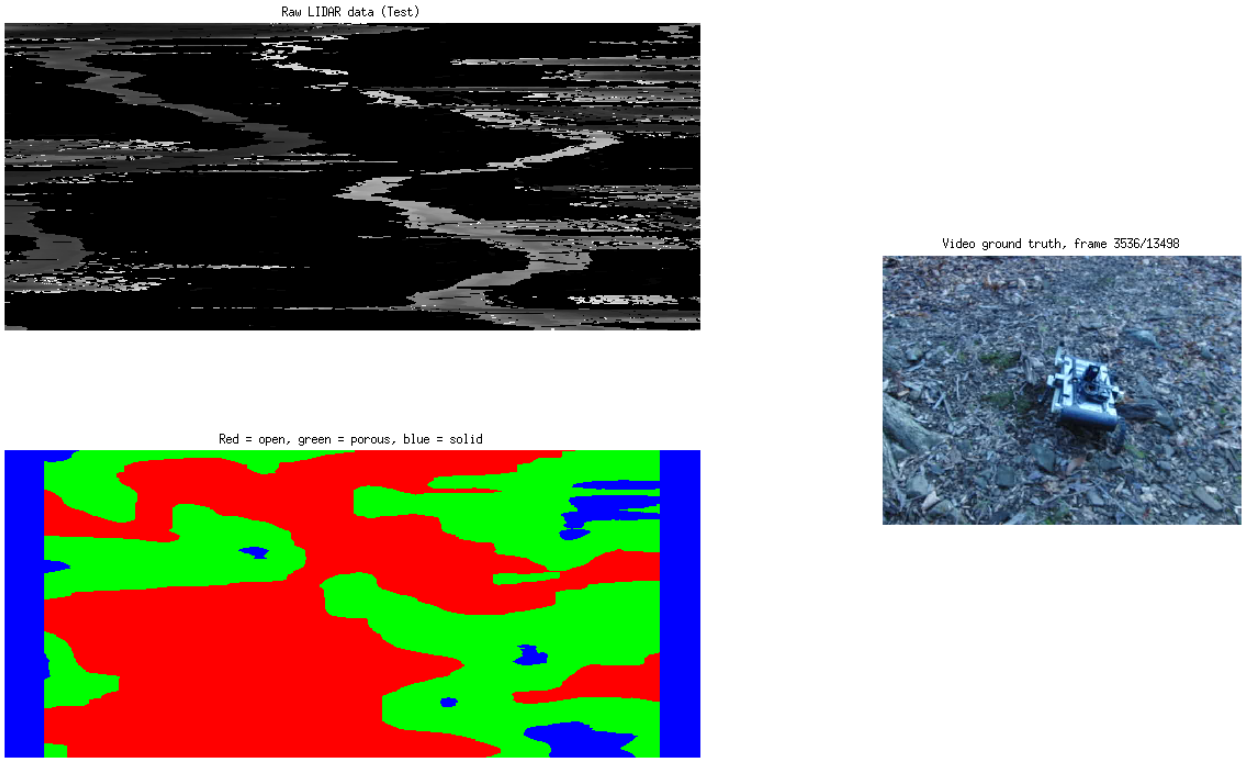

Figure 5: This figure illustrates the performance of the rank-order statistic, which performed very similarly to the L_2 norm method but tended to classify open areas and solid obstacles as porous obstacles more readily. Here the robot will shortly be negotiating two trees, which involves a lot of servoing back and forth. This is the worst case encountered in the whole data collection sequence. First, the trees are mostly classified as porous obstacles, where they should be solid. Second, each of the trees is lost during a servoing maneuver.

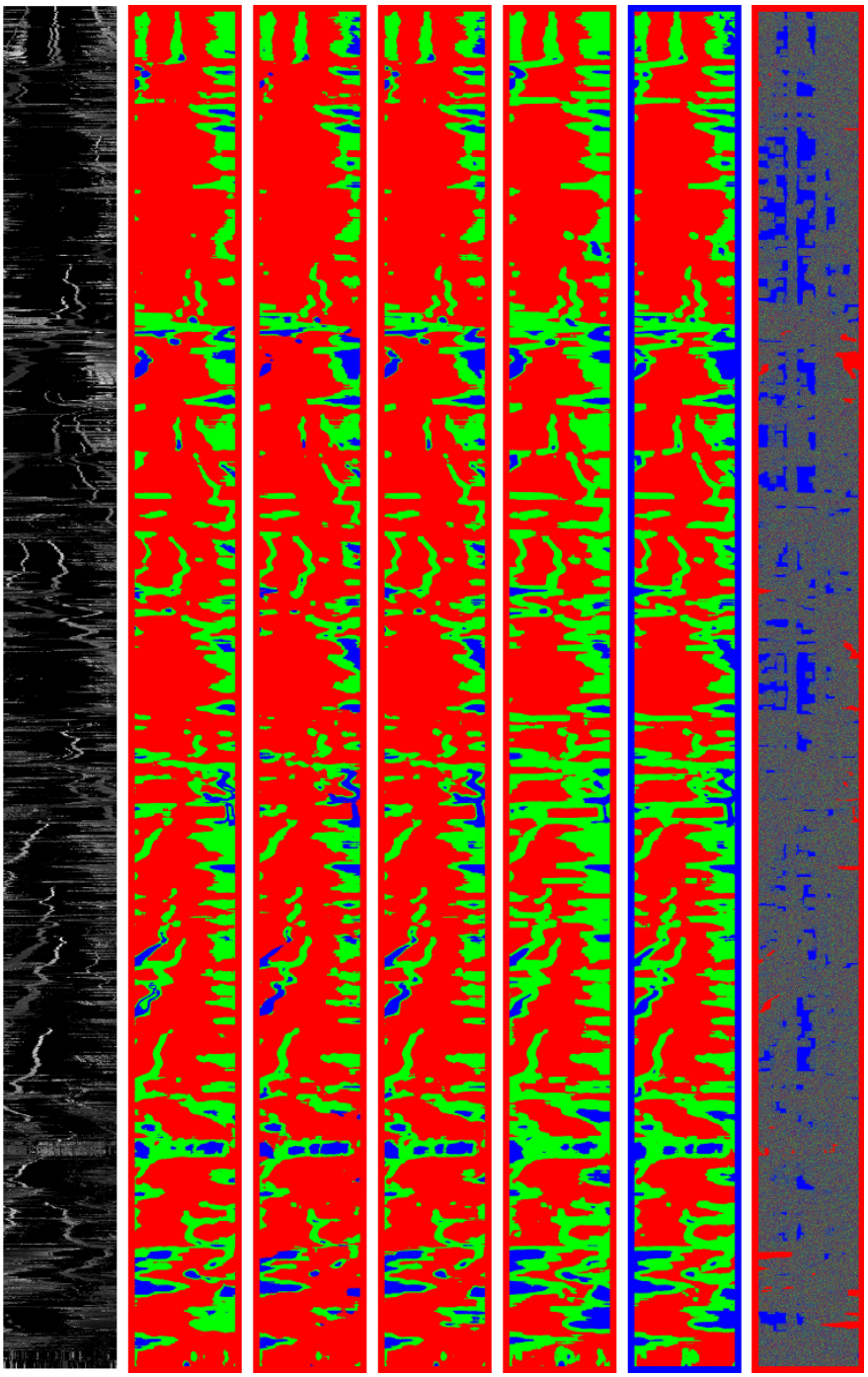

Figure 6: This figure provides comparisons of the different distance measuring methods for the nearest-neighbor classification scheme used in the preliminary work done in the Learning in Robotics class. Notice that the Kolmogorov distance and L_p norms provide very similar classifications, while the Kullback-Leibler divergence and the rank-order statistic also look very similar. The classifications provied by the bounded L_∞ norm closeness test are not random, but are also not reliable. For comparison, I also included the raw LIDAR scan. In order from left to right, the classification schemes are Kolmogorov, L_1, L_2, Kullback-Leibler, the rank-order statistic, and the bounded L_∞ norm closeness test.