Classification of 3D Object Scans in Urban Environments

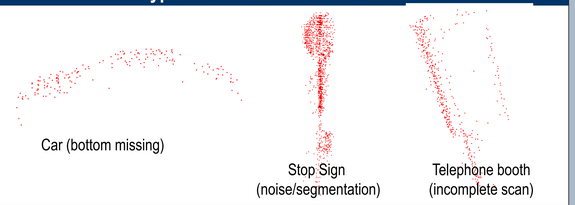
Christopher Tralie. ctralie@princeton.edu
Class of 2011 Electrical Engineering, Princeton University
Thomas A. Funkhouser funk@cs.princeton.edu
Professor of Computer Science, Princeton University

Purpose

The purpose of this project is to explore classification of 3D point cloud blobs extracted from a scan of a city into different object categories. We start with an enormous data set: a 3D scan of the city of Ottawa, merged from laser scans of the city. Work has been done already to segment this enormous point cloud into smaller objects of interest, the "blobs," which are denser collections of points separated from the background that likely make up one continuous object (e.g. a fire hydrant, a car, or a mailbox). Some work has also been done classifying these objects using results from the segmentation in addition to some 3D shape descriptors.

My main work this semester expands the classification techniques of these segmented blobs by adding a novel 3D shape descriptor and analyzing the extent to which it improves classification. The main challenge of this research comes from incomplete, unstructured object scans due to high noise (from the scanner and improper segmentation) and occlusion (due to the limited viewpoint of the scanner). The new descriptor, which tries to align these point cloud scans to known models of city objects in a database, is designed to be resistant to noise and especially to occlusion.

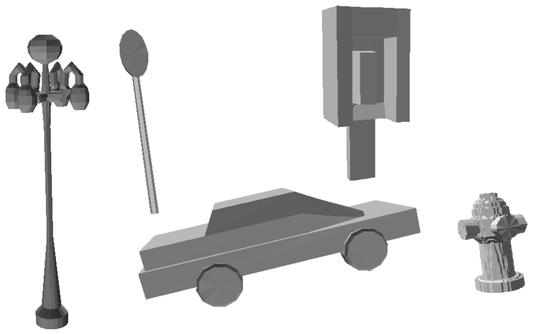
Typical Point Cloud "Blob" Scans



Car (bottom missing) | Stop Sign (noise/segmentation) | Telephone booth (incomplete scan)

Feature Vector from the Mesh Database

There are 42 hand-modeled triangular meshes of likely city objects. A 3D scan is aligned to each of these 42 models and 42-D feature vector is computed, where each component is the fraction of points that fall within a certain distance **eps** of the 3D object from the mesh. This ensures that partial scans of a model still get a good score with that model



A = 1_advertising_cylinder
C = 12_dumpster
D = 14_fire_hydrant
E = 15_flagpole
F = 24_mailing_box(free_standing)
G = 26_newspaper_box
H = 29_parking_meter
I = 33_recycle_bins
J = 43_trashcan
K = 75_highway_sign
L = 97_telephone_booth
M = 133_traffic_control_box
N = 134_traffic_control_unit
O = 138_lamp_post_one_bulb
P = 140_lamp_post_three_bulbs
Q = 141_lamp_post_four_bulbs
R = 142_lamp_post_five_balls
S = 143_lamp_post_three_and_two_balls
T = 144_lamp_post_tall
U = 145_lamp_post_on_light_standard
W = 150_stop_sign
X = 152_street_name_sign
Y = 153_other_traffic_sign
Z = 154_tall_thin_sign_on_pole
AA = 157_a_frame_sign_on_ground
AB = 161_traffic_light_no_arm
AC = 162_traffic_light_half_arm
AD = 163_traffic_light_one_arm
AE = 164_traffic_light_mult_arm
AF = 165_traffic_light_on_light_standard
AG = 172_short_post_in_row
AH = 175_tall_post_in_fence
AI = 181_light_standard_no_arm
AJ = 182_light_standard_mid_arm
AK = 184_light_staandard_T_with_sign
AL =
185_light_standard_no_arm_with_sign
AM =
186_light_standard_top_arm_with_sign
AN =
187_light_standard_mid_arm_with_flag
AO = 191_car_sedan
AP = 192_car_van
AQ = 193_car_pickup
AR = 194_car_truck

Alignment Using Iterative Closest Points (ICP)

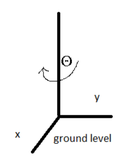
Iterative Closest Points (ICP) is a known technique to bring different objects close to each other using some distance metric. In this case, a point cloud is aligned to a mesh model using Euclidean distance, where distance is calculated between each point and the closest point on the mesh (accelerated with a spatially partitioned mesh search tree). At each step, an optimal transformation is calculated to bring the point cloud as close as possible to the mesh with this overall distance metric between all of the points. Then another optimal transformation can be calculated, and this process is repeated until convergence. In this manner, the point cloud "snaps" into place at some local minimum of Euclidean distance (like many iterative techniques, it isn't guaranteed to have a global min). Note also that good initial alignments with Principal Component Analysis techniques (using SVD) can help the convergence

3DOF vs 6DOF Alignment

At each step of ICP some optimal transformation is calculated to bring the points closer to the mesh. Initially, we optimized with a 6D affine transformation: 3 degrees of freedom for (x, y, z) translation and 3 degrees of freedom for any rotational orientation. But a more physically sensible choice may be to only to allow xy translation on the ground (to prevent objects from being "lifted") and rotation about the z-axis, with 3 degrees of freedom total. Both of these techniques are tested and compared to each other, with the hypothesis that 3D will outperform 6D since it makes more physical sense

6DOF

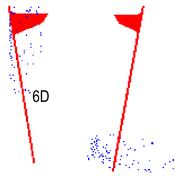
$$\begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



3DOF

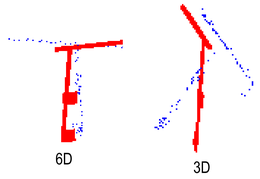
$$\begin{bmatrix} r_{11} & r_{12} & 0 & t_x \\ r_{21} & r_{22} & 0 & t_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Car to flagpole



This example shows the potential benefits of using 3D over 6D. The car is nothing like the flagpole, but the 6D alignment rotates the car into a physically impossible position suspended in the air facing down where it matches the flagpole rather well. The 3D alignment, on the other hand, preserves the car at ground level where few points are close to the flagpole

Highway sign to highway sign

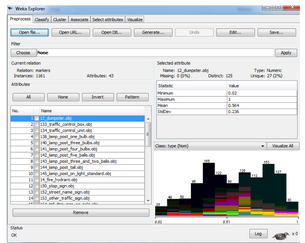


This example shows the potential dangers of using 3D over 6D. The highway sings are bent slightly different ways. The 6D is able to fix this, but the 3D cannot, so it gets a low score even though it is itself a highway sign

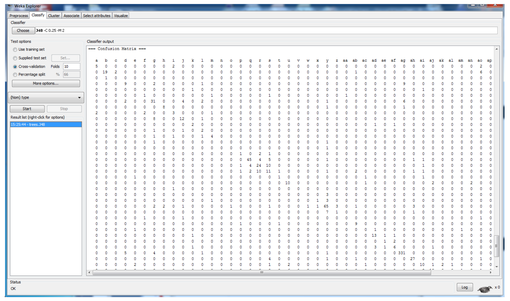
Batch Testing and Machine Learning with Weka

To test the accuracy of this new feature, I started with 1163 object scan segmentations from the city as training data. These objects had all been classified by hand ahead of time so I knew the type. I then computed the 42-D feature vector by matching each scan up with the objects in the database and computing the ratio of the points that fell within some distance **eps** of the mesh after alignment. I did this for all 1163 objects first using 6DOF alignment and then using 3DOF alignment. I also varied the cutoff distance distance **eps** between 0.05m, 0.1m, 0.2m, 0.4m, and 0.8m. The idea here was that a smaller eps gives lower scores because alignments have to be more exact for the points to be that close, but it may be too harsh (so there's some trade-off that I wanted to assess).

Once all of the feature vectors were computed varying the different parameters, an open source machine learning program called "Weka" was used to test the effectiveness of the classifiers. I chose to classify the objects using a J48 decision tree built off of the training data. 10 "folds" were used, which means that the 10 sections of 10% of the training data were taken out and tested against the remaining 90%. Here are a few screenshots of Weka in action.

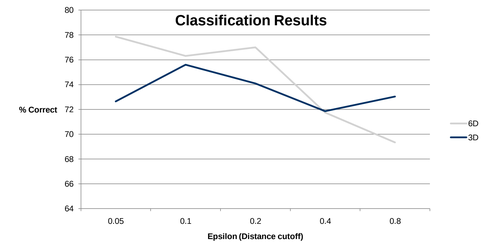


Weka showing the histogram of one of the distance features, which is on the interval [0, 1] since it is the ratio of points that fell within a certain distance epsilon of the model associated with that feature



Weka showing a **confusion matrix** of my classifier in one of the tests. A confusion matrix has the known object type in the rows, and the guesses of other class types in the columns. Entries along the diagonal are correct guesses. Entries off the diagonal are mistakes.

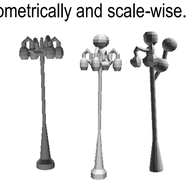
Results / Conclusions



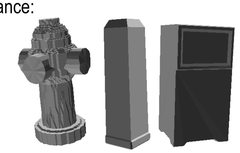
Classification was overall successful for both 6D and 3D alignment. Optimal epsilon for 3D was about 0.1, while the optimal epsilon for 6D was 0.05. These small epsilon values likely outperformed the larger ones because they were more discriminative.
Surprisingly, 6D outperformed 3D overall, perhaps because of this particular data set. More analysis will be done before the final writeup as to why that is the case

Common Confusions

Most of the confusions make sense and are of very similar objects geometrically and scale-wise. For instance:



Lamp posts with different numbers of bulbs were often confused with each other



Fire hydrants, "short posts," and mailboxes were confused with each other, presumably because the scans were not of fine enough resolution to pick out the more subtle details (and these objects are all of similar size)

References

[1] Boyko, Aleksey. "Lidar City Project instructions." *Computer Science Department at Princeton University*. Web. 26 Feb. 2010. <http://www.cs.princeton.edu/~aboyko/city_doc/mainInstructions/>.
[2] Funkhouser, Thomas, Aleksey Govolinsky, and Vladimir G. Kim. "Shape-based Recognition of 3D Point Clouds in Urban Environments." Print.
[3] Funkhouser, Thomas, and Aleksey Golovinsky. "Min-Cut Based Segmentation of Point Clouds." Print.
[4] Funkhouser, Thomas, Patrick Min, Michael Kazhdan, Joyce Chen, Alex Halderman, David Dobkin, and David Jacobs. "A Search Engine for 3D Models." Print.
[5] Yaroslav Halchenko, Yaroslav. "Iterative Closest Point (ICP) Algorithm. L1 solution. . ." Web.

Acknowledgements

•**Tom Funkhouser**: Faculty adviser, provided codebase, algorithm ideas/advice, and extreme debugging help
•**Aleksey Boyko**: Graduate student in computer science, started city project and advised me on many technical points along the way