Josh Beal, Gabe Alvarez

Detection of Landable Areas for Aerial Vehicles

Abstract

Determining the optimal drop-off location within a property is an important step in developing an autonomous delivery system using an unmanned aerial vehicle. We can use supervised machine learning algorithms to develop a notion of what reasonable drop-off locations look like, based on labeled satellite images. We have reached a strong predictive baseline for drop-off areas using an SVM that makes predictions from the color histogram and edge density of image patches, and have designed a pipeline that can then eliminate problem objects and identify a single landable point for the vehicle.

Introduction

A currently ongoing task in robotics is the quest to create an unmanned aerial vehicle that can make deliveries autonomously. Although there are many aspects to this complex task, one hurdle that must be overcome is teaching the UAV where on a property to deliver a package based on the address of that property. For example, it would be desirable to drop a package on a person's doorstep or in their yard, but not on the roof of their house or into a tree.

Knowing the address of a drop-off location, we can use Google Maps to acquire a satellite image of the property, but this does not tell us what pieces of the property are reasonable drop-off locations. In this paper, we develop a supervised machine learning algorithm that processes satellite images of residences to determine which areas of the image represent reasonable drop-off locations, and then from there derive a single pixel that represents an optimal drop-off location.

Data Collection

In order to arrive at test and training satellite images, our first task was to arrive at a list of valid addresses. We used a service known as Maponics to pay for a large list of addresses in the Palo Alto area. From this list, we used the Google Static Maps API to acquire a satellite image for each address, and checked the SHA-256 hashes of the images to remove duplicates.

For our supervised learning, we labeled our data in an image editor by copying each image and editing the copy so that all landable pixels had the same color, one which never appears naturally in satellite images of houses, such as #FF0FF. We then computed feature vectors from the processed image data.

Features

Our feature set went through a large amount of incremental development over the course of our project. Our first attempt at supervised learning used single pixels as training examples and ran an SVM on a subset of the labeled pixels to arrive at a result. Due to inaccurate results from this approach and excessive convergence time for our SVM algorithm, we decided to re-work our overall strategy and use small square patches of an image as training examples instead of individual pixels. We were able to compute these new labels from our previous individual-pixel labels by labeling a patch as landable if it had a number of landable pixels greater than some threshold value. After some cross-validation to determine which threshold would allow our algorithm to best distinguish between patches, we decided to label a patch as landable if 45% or more of its pixels were landable.

Our first feature set, which used pixel color exclusively, performed similarly well on both

our test and training sets, so we saw that our algorithm could likely be improved by taking steps to reduce bias. We combated this by adding in more features that take into account different aspects of the image. In particular, adding features that captured the context of a pixel contributed to increases in performance, as telling whether an area is landable depends more on what is around it than its color alone. Indeed, two pixels of the exact same color could be part of a house's roof in one case and part of a walkway in another, and information beyond simply that pixel alone is necessary to distinguish the two.

For our final feature set, we used measures of both the color and texture of a 10x10 square patch of pixels to make predictions. Color features consisted of the average red, green, and blue values for the patch, as well as color histograms for red, green, and blue with 10 buckets each. These color histograms stored what percentage of the pixels in the patch had a color value in a given range (for example, how many pixels in the patch had a blue value between 230 and 255).

We computed our texture features using a measure of edge density derived from the Canny edge detection algorithm. We first ran the Canny algorithm on an image, and then for each patch we computed its edge density feature by calculating the percentage of pixels in that patch which were edges. We derived further texture features for a patch by computing the edge densities of areas surrounding the patch. Our intuition here was that edge density would be useful in determining a patch's overall context in the image; for example, areas of an image corresponding to a tree are likely to have very "noisy" surroundings, whereas a driveway or path to a house is likely to be more "quiet," and hence our algorithm would be able to more easily distinguish between the two.

The exact numbers to use for many aspects of our features were unclear to us at first, and we arrived at our final values using cross-validation. For example, our patch sizes, number of buckets in our histograms, and number of texture features were all derived in this way.

SVM Algorithm

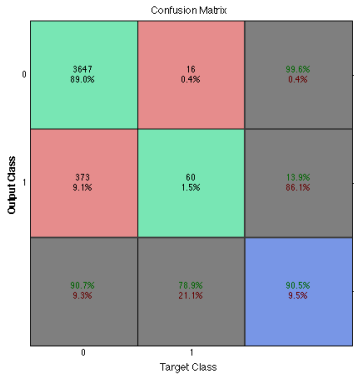
We make predictions of landable patches via an SVM algorithm using the libsvm package. We developed a script that converts from an image format to a list of patches and feature vectors, to be used as input to libsvm, and another script that converts back from an output file generated by libsvm to an image. Viewing our output as an image is helpful to determine what types of mistakes our algorithm is making, and where it could be improved.

The specific parameters for our algorithm have gone through a number of improvements over the course of our project. Our initial attempt at an SVM algorithm consisted of using a Gaussian kernel with default options provided by the libsvm interface to MATLAB. This approach had two distinct problems, which we took different steps to remedy. First, our initial algorithm was making almost no positive predictions. We determined that this was because our classes were skewed toward negative examples, and so we decided to run an SVM that weighted the positive examples more heavily in our training set. After some cross-validation, we decided to weight our positive examples 3 times as heavily. Second, when run on our full feature set, our algorithm seemed to be overfitting our data, which we concluded after testing on our training set and observing 99.9% accuracy, as compared to much lower (88%) accuracy on our cross-validation test set. To combat this overfitting, we changed our kernel to a linear kernel and decreased our regularization parameter from 1 to 0.0001 and observed improved predictive accuracy.

Our current algorithm consists of running a C-SVM with a linear kernel, regularization parameter C = 0.0001, and positive examples weighted x3.

Neural Networks

Based on Volodymyr Mnih's recent Ph.D thesis on "Machine Learning for Aerial Image Labeling", we saw potential for using neural networks to learn discriminative features in aerial images. While his work had success with roads and buildings, we hoped to extend it to identify landable areas. We implemented a feed forward neural network in MATLAB using the features described earlier, which we saw predicted too few landable areas. Larger training sets may enhance performance, as the current network is likely over-conservative due to the small number of landable examples observed.

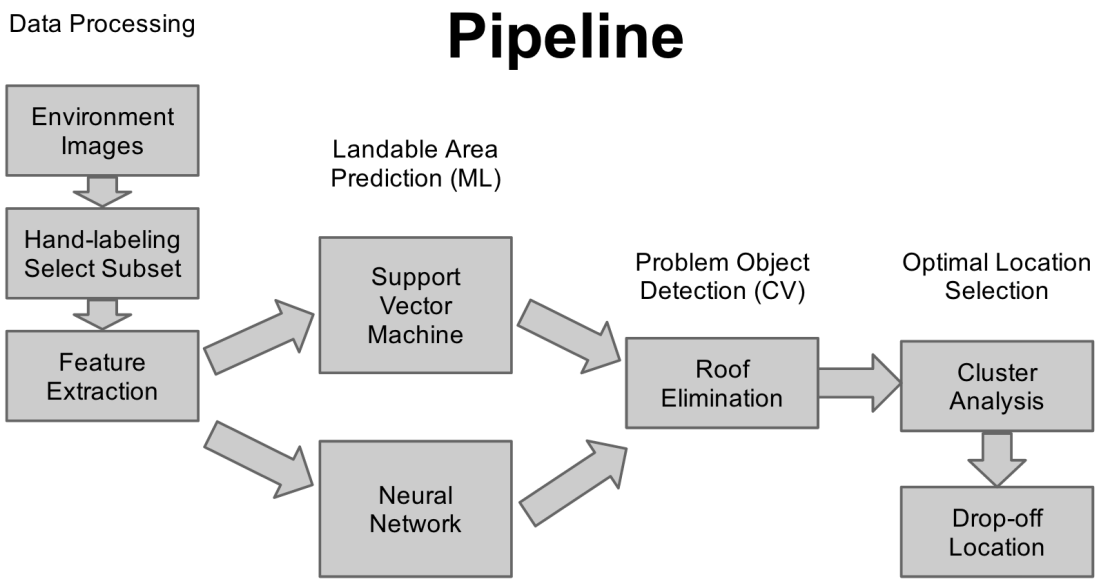


Results

With a 48-dimensional feature vector consisting of color and texture features, our algorithm was able to attain cross-validation test error of 9.5%. While some landable areas were not predicted as landable, for our application it is most important to avoid false positives, as false negatives are less likely to be the ideal landing location. As the confusion matrix above shows, 78.9% of landable predictions were correct. Moreover, many of the false positives fit into object classes such as building roofs that can be easily detected using techniques from computer vision.

Fig 1. Labeled Example

Fig 2. Classification Result of SVM



Pipeline & Future Directions

The machine learning algorithm that we have developed for predicting landable patches of an image fits into a larger pipeline of steps to arrive at a single landing site from a satellite image. Our algorithm is able to arrive at a number of candidate patches which are either actual landable areas or building roofs with high likelihood. The next step in this pipeline is to remove roof patches from our list of candidate patches by employing a computer vision based roof detection algorithm. Such algorithms have already been developed; a paper detailing one such algorithm can be found at http://www.cis.rit.edu/~sxs4643/igarss2012.pdf.

Once roof patches have been eliminated, it remains to arrive at a single result patch from the set of candidates. We have implemented an algorithm that takes makes this decision by finding the largest clusters of candidate pixels in the image and choosing a pixel in one such cluster; the intuition here is that the best landing site is likely part of a large landable area. One could likely achieve more accurate results using a more nuanced algorithm that takes into account the confidence of the SVM's predictions on each pixel as well as cluster size.

In the future, we could extend the functionality of our algorithm to diverse data sets, such as street view images or a real-time feed from the quadcopter. This data is likely to introduce new difficulties regarding the usefulness of color features, as camera type may introduce a different color palette. For this reason, additional texture-based features are likely to be important in obtaining accurate predictions.

With the announcement of Amazon Prime Air, excitement is growing for quadcopter-powered deliveries, and our work can play a critical role in enabling better drop-off location prediction and safer deliveries. In the future we hope to use this system in partnership with Manna, the autonomous quadcopter project under development by Josh Beal's team at the Stanford UAV Lab. Both machine learning and computer vision will play an important role in the future of our transportation system, and extending this work to real-time production systems could lead to greater reliability and safety.