# Orientation-Aware Scene Understanding for Mobile Cameras

**Jing Wang**
Georgia Inst. of Technology
Atlanta, Georgia, USA
jwang302@gatech.edu

**Grant Schindler**
Georgia Inst. of Technology
Atlanta, Georgia, USA
schindler@gatech.edu

**Irfan Essa**
Georgia Inst. of Technology
Atlanta, Georgia, USA
irfan@cc.gatech.edu

## ABSTRACT

We present a novel approach that allows anyone to quickly teach their smartphone how to understand the visual world around them. We achieve this visual scene understanding by leveraging a camera-phone's inertial sensors to lead to both a faster and more accurate automatic labeling of the regions of an image into semantic classes (e.g. sky, tree, building). We focus on letting a user train our system from scratch while out in the real world by annotating image regions in situ as training images are captured on a mobile device, making it possible to recognize new environments and new semantic classes on the fly. We show that our approach outperforms existing methods, while at the same time performing data collection, annotation, feature extraction, and image segment classification all on the same mobile device.

### Author Keywords

Visual scene understanding, mobile phone cameras, orientation-aware, active learning.

### ACM Classification Keywords

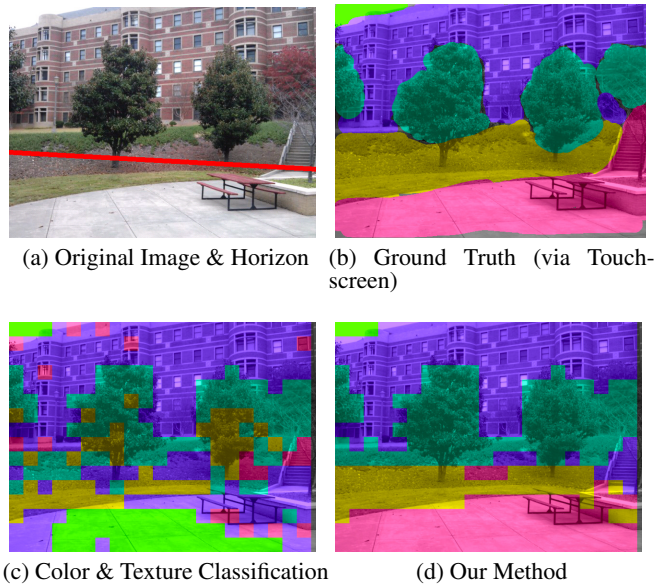
H.5.2 Information interfaces and presentation (e.g., HCI): Miscellaneous.

### General Terms

Algorithms, Design, Experimentation.

## INTRODUCTION

Cameras are becoming ubiquitous, especially as they merge with smartphones. This merger of smartphones and cameras has led to an exponential increase in image capture, sharing, and all other related applications [9]. With this merger, cameras now have access to a variety of additional rich information, as most smartphones are equipped with an array of specialized sensors. Specifically, smartphones come with accelerometers, gyros, and magnetometers that enable a continuously updated estimate of the global rotation of the camera with respect to a world coordinate frame. Furthermore, the user can also directly add information by tagging and annotation right at capture. We seek to use this information to help with scene understanding via images captured by these mobile cameras.



(a) Original Image & Horizon (b) Ground Truth (via Touchscreen)

(c) Color & Texture Classification (d) Our Method

Figure 1: To aid scene understanding, (a) inertial sensors estimate the horizon in images captured by a smartphone camera. (b) Users provide ground truth annotations via smartphone touchscreen. (c) Approaches based on color and texture show confusion between grass and trees, and between sky and ground. (d) Using horizon information, our method solves such problems. The entire system, including data capture, annotation, training, and testing, runs on a smartphone out in the real world.

The field of computer vision has sought answers to questions of understanding scenes for various applications ranging from object and scene categorization to navigation and planning, purely by image analysis. We propose to combine such image analysis with additional information available from smart phone sensors and from the user of the camera. Some computer vision techniques are just beginning to make serious use of the data from these non-visual sensors as a core part of vision algorithms [6, 7, 10]. However, the task of scene understanding – attempting to identify and associate a class label to every pixel in an image – has not yet been


*UbiComp '12*, Sep 5-Sep 8, 2012, Pittsburgh, USA.

tackled in the light of this additional sensory data. We propose a scene understanding approach that takes advantage of this orientation information to improve image annotation and the related scene and object class labeling accuracy.

As noted above, automated scene understanding has traditionally been important in computer vision research, as it is useful for content-based image retrieval and annotation. Furthermore, it has value in robotics, because a robot must understand the visual world in order to interact with it and navigate through it. Both these applications for image annotation and navigation are also of value in mobile and ubiquitous computing, leveraging the popularity and applicability of mobile cameraphones. The system and algorithms we present will have a significant impact in these problem domains, especially as home robots become more commonplace and users need to teach them about their environment.

Many previous approaches to scene understanding have introduced algorithmic complexity to deal with issues of context and relative location of objects [3, 15]. While such methods are valuable, we show that knowledge of camera orientation alone is capable of resolving a number of the ambiguities that such methods are intended to overcome (e.g. the fact that sky appears above trees while grass appears below on the ground). An added benefit of this sensor-based approach is that it reduces the need for expensive computation regarding context and enables us to move scene understanding onto mobile devices with limited computational power.

Not only do we exploit the sensors on mobile devices, but we demonstrate that modern smartphones are now capable of performing every step in constructing a scene understanding system: image capture, ground truth annotation, feature extraction, and classification of new images. In contrast, most approaches to scene understanding treat the task as an offline process to be performed on a desktop computer after collecting images from the internet. The primary disadvantage to this traditional approach is that any information not embedded in the EXIF tags of an image is lost. We solve this problem by capturing all images, both the training and testing sets, on a mobile device which records information from the device's sensors. A further advantage of our approach is that it closes the loop of data collection, training, and testing for a scene understanding system. An individual with a smartphone can train the system from scratch while out in the real world, collecting and labeling additional training data as needed to improve system performance. A similar approach to recognizing objects on mobile devices [11] has found success recently.

Following the approaches of [12] and [16], we adopt a non-parametric scene parsing method that finds matching image segments in query and database images, transferring the label of the database segment to the query segment. Furthermore, we account for camera orientation by identifying each image segment as *above*, *on*, or *below* the horizon in the image, providing physically grounded context for the scene. We explore several methods of incorporating this horizon information into scene parsing and show that our orientation-aware approach produces marked improvements in labeling accuracy.

The primary contributions of this work are (1) *an orientation-aware scene understanding method* and (2) a *fully mobile scene understanding pipeline which exploits smartphone sensors and which we are releasing for others to use*.

## RELATED WORK

Recent work on scene understanding has attempted to address the issue of context and to model the location of different object classes across images. Approaches have included discriminative semantic segmentation methods that account for context [15] and multi-class image segmentation approaches with relative location priors [3, 2] to model the spatial relationships between classes. The earlier work of [17] introduced contextual priors for object localization based on semantic scene categorization. In contrast to these computationally expensive solutions, we introduce a feature, based on position with respect to the horizon, which solves many of the same problems with minimal additional computation.

Similarly, [16] uses the vertical position of an image segment within the image as a feature. Since images vary in orientation and field of view, the pixel position within the image is not really what is important. For this reason, implicit assumptions are usually made about camera orientation which introduces a bias to the images used in databases for scene parsing. Since on a mobile device we have access to absolute image orientation, we instead use the position of an image segment with respect to the horizon as a feature. This is a more meaningful and physically grounded measurement.

The idea that scene understanding can be improved with knowledge of the camera's viewpoint has been explored by Hoeim et al. [4]. Their method performs horizon and camera height estimation to improve pedestrian and car detection performance on images from the LabelMe dataset [14], simultaneously refining horizon and height estimates based on detection results as well. Gould et al. [2, 3] have also presented a method that incorporates a measure of position with respect to the horizon into scene understanding. However, these methods and others [5] make the simplifying assumption that the horizon is horizontal in the image, and that the image is captured from a mostly upright viewing angle. Our method places no restrictions on camera orientation, and can therefore deal with a much broader variety of images as well.

Early examples of the fusion of visual and inertial sensors exist in the robotics and augmented reality literature [18, 19] where online visual feature tracking has been used to prevent drift in pose estimation from inertial measurements. The structure from motion approach of [8] also incorporates inertial measurements to improve performance.

We use accelerometers, gyros, and magnetometers to obtain camera orientation for scene understanding, while several recent computer vision approaches have used inertial measure-

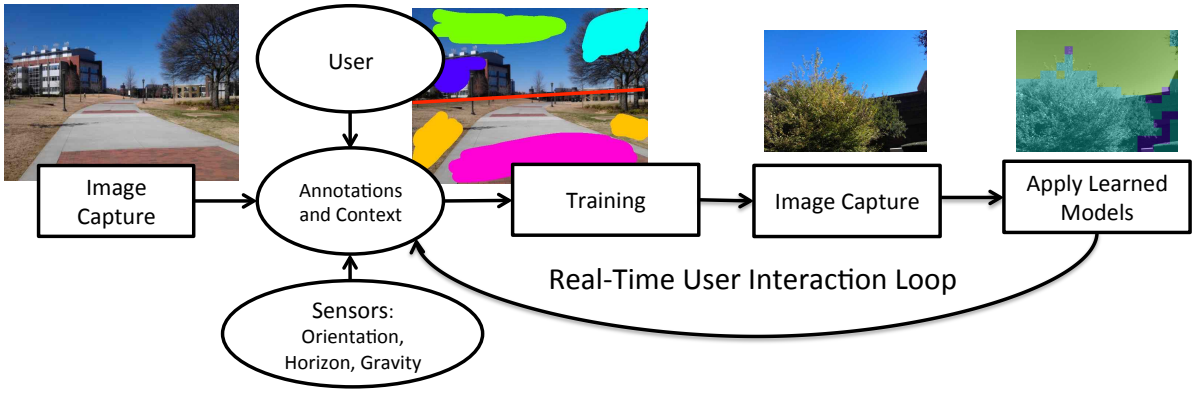



Figure 2: System Diagram. We present the first end-to-end system capable of training and running a scene understanding system on a mobile device, which involves: capturing images and other sensor data, letting a user annotate the image, training an image segment classifier based on user annotations while still out in the field, and applying the learned models to newly captured images so that the user is able to see how well the system is performing. This loop then iterates until the user is satisfied with the system's scene understanding performance.

ments to perform a variety of other tasks. In [10], detected SIFT features are aligned with the gravity vector in order to remain rotation-invariant. The work of [6, 7] improves KLT feature tracking performance in a moving camera by integrating inertial measurements as well. As in [10], we find that incorporating camera orientation improves both computational efficiency and classification accuracy.

## SCENE UNDERSTANDING ON A MOBILE DEVICE

An important feature of our mobile scene understanding approach is the fact that the entire system not only runs on a smartphone, but can be trained and tested, starting from scratch, on a smartphone (see Figure 2). The important steps include:

**Data Collection:** Images along with their orientation and horizon information are collected using the smartphone camera to build up a database.

**Image Annotation** Database images are annotated with ground truth class labels via the smartphone's touchscreen. Entirely new classes can be added at any time.

**Training** Modern smartphone CPUs are capable of performing all necessary feature extraction onboard. Feature extraction and image annotation are all that is required to add new training data to the system.

**Testing** Newly captured images are divided into segments, each of which is classified via matching to labeled database image segments. After observing classification performance on a newly captured image, a user may choose to annotate any incorrectly classified areas of the image in order to improve performance.

This closed loop opens the door to on-the-fly collection and annotation of new training data in order to improve the system's performance while out in the real world. We are releasing the system to the research community as an Android app with source code available at the following address: *http://code.google.com/p/gatech-mobile-scene-understanding*.

## HORIZON-AWARE SCENE UNDERSTANDING

We perform scene understanding on a mobile device with the aim of providing a class label for every pixel in an image (see Figure 1). One of our primary contributions is that we incorporate global camera orientation into a nonparametric scene parsing method using a new feature based on the position of the horizon in the image. We first provide a brief overview of the basic scene understanding approach we have adopted before explaining the details of our method.

### Background: Nonparametric Scene Parsing

Given a database of images annotated with ground truth class labels at every pixel, we wish to label the pixels in a new query image. Rather than training a discriminative classifier [15], recent nonparametric approaches [12, 16] have matched image segments $S_{query}$ in a query image to similar image segments $S_{database}$ in a training database in order to transfer the labels from the database to the query image segments. In the approach of [16], each image is divided into a number of segments, and a variety of features, including those based on color, texture and image location, are extracted from each segment to be used for matching. For the sake of efficiency and also to improve accuracy in [12] and [16], a global image matching step narrows the database to a small *retrieval set* of images that are globally similar to the query image. Finally, each segment in the query image is compared to all segments $S_{retrieval}$ in the *retrieval set* in order to choose the segments from which labels will be transferred.

The advantage of this nonparametric approach is that there is zero training time for the system other than feature extraction for each image segment in the database. *This means the database can grow over time without any need for retraining a classifier, which is crucial for our on-the-fly data collection approach.*

### The Horizon as Context

We use knowledge of the horizon's position in an image to provide a form of context that is grounded in the physi-

Figure 3: Horizon from Camera Orientation: Arbitrary camera rotations are not a problem for our scene understanding approach. Even if the horizon is not visible in an image, we know which parts of the image are above the horizon and which parts are below the horizon.

cal world's coordinate system rather than in image coordinates. In contrast, modern scene understanding approaches [3, 15, 16] overwhelmingly model context in image coordinates. While several methods [5, 2] attempt to estimate the horizon at query time, the requirement that the horizon be both visible and completely horizontal rules out these approaches for many real world images. Moreover, there is computational complexity involved in traditional context models that rely on conditional random fields (CRFs) and additional spatial relationship priors.

We observe that some of the complexity of context-based approaches, grounded in image coordinates, can be removed when we are able to measure the true position of the horizon in the image. We are able to learn not only that the sky appears above the ground, but that sky appears *above the horizon* while the ground appears *below the horizon*. Thus, for example, we shouldn't expect to see the ground at all when the camera is pointing up.

#### Horizon from Camera Orientation

We use the measured orientation of the camera to determine the location of the horizon in any image (see Figure 3). Equipped with accelerometers, gyros, and compasses, modern smartphones are able to report their absolute orientation in the world coordinate frame as a $3x3$ rotation matrix $R$. Assuming a known focal length $f$ and camera center $(u,v)$, we project distant points on the ground plane into the image and join them to determine the horizon line $h$. We now explain this process in more detail.

Knowing the orientation of the camera is what will allow us to determine the location of the horizon in each image, so we must first determine a rotation matrix $R$ that describes how the camera is oriented in the world. Because the three columns $(v_1, v_2, v_3)$ of all rotation matrices are mutually orthogonal vectors, one can construct a rotation matrix given any two orthogonal vectors (e.g. $v_2$ and $v_3$) that describe the axes of the rotated coordinate system, with the remaining column computed as the cross product of the two given vectors ($v_1 = v_2 \times v_3$). In our case, a 3-axis accelerometer provides a gravity vector as $v_3$ while a 3-axis magnetometer provides another vector $v_2$ pointing toward magnetic north. In fact, modern Android devices come equipped with built-in methods to compute the rotation matrix $R$ from available sensor data in a similar manner.

Next, we combine the rotation matrix $R$ with additional information about the camera to determine a 2D line $h$ that describes the horizon in an image. We do this by projecting into the image a number of 3D points which lie on the ground plane. By the standard equations of perspective projection, a 3D point $P = (X,Y,Z)$ in the world projects down to a 2D point $p = (x,y)$ in the image as:

$$x = f\frac{X}{Z} + u_0 \qquad y = f\frac{Y}{Z} + v_0$$

with the camera's focal length $f$, and center of projection or principal point $(u,v)$ expressed in pixels. The focal length $f$ is often stored in the EXIF tags describing the properties of an image, while it is common to assume the center of projection resides in the geometric center of the image, with $u_0 = w/2$ and $v_0 = h/2$ for an image of width $w$ and height $h$ in pixels. We generate a number $n$ of 3D points in a circle on the ground plane $P_{1..n} = \{k\cos(2\pi\frac{i}{n}), k\sin(2\pi\frac{i}{n}), 0\}_{i=1}^{n}$ for a large value of $k$ to put the 3D points sufficiently far from the camera to appear on the horizon. We rotate these points into the camera's coordinate frame by multiplying each point by the rotation matrix $R$, and then project these rotated 3D points into the 2D image using the standard projection equations above. Finally, we find the two projected points with the minimum and maximum x-coordinates that still lie within the horizontal field of view of the camera (though they may not actually be visible in the image) and treat these as the endpoints of the horizon line $h$.

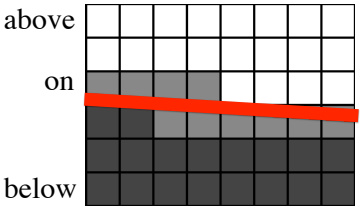



Figure 4: Horizon-Based Features: Each image segment is designated as being *above*, *on*, or *below* the horizon, a feature grounded in the physical world rather than in image coordinates. We combine this horizon feature with color and texture statistics to transfer class labels between similar image segments.

In addition to the horizon line $h$ itself, we also retain knowledge of the direction of gravity (i.e. which side of the horizon is *up* and which is *down*). With this knowledge, an advantage of our approach over all previous approaches is that we don't need to worry about dealing with landscape, portrait, or arbitrary orientations of images. Such issues are problematic for any method which takes into account the absolute or relative position of segments in image coordinates. Our method is able to cope with completely arbitrary camera rotations, even if the horizon itself is not visible in the image (e.g. images of ground or sky) as in Figure 3.

#### Horizon-Based Features

To make use of horizon information, we assign to each image segment $s_i$ a position label $p_i \in \{above, on, below\}$ which identifies each image segment as depicting a part of the world

that is *above*, *on*, or *below* the horizon (see Figure 4). To do so, we compute a bounding box around every image segment and then determine the position of each corner of the bounding box with respect to the horizon. If all four corners of the bounding box lie above or below the horizon, the entire segment is classified as *above* or *below* the horizon, respectively. Otherwise, the segment is classified as *on* the horizon.

### Color and Texture Features

We also compute color and texture features for each image segment. This gives us a baseline feature set for classification of new images, to which we will add horizon information in order to improve performance. The color information is a histogram of RGB colors (60 dimensions, made up of 20 bins per color channel) and the texture information is a histogram of 100 visual words, which are quantized dense SURF features [1] detected every 3 pixels within an image segment. When matching query image segments to database image segments, we maximize histogram intersection across these color and texture features.

### Combining Horizon with Color and Texture

Given the known horizon in an image, we explore multiple ways to incorporate this knowledge into our scene understanding approach, aiming for computationally efficient approaches.

#### *Horizon-Based Retrieval Sets (HBRS)*

Our first method works by dividing our training data into three sets: a set of image segments $S_{above}$ that appear above the horizon in the training images, a set of image segments $S_{on}$ that span the horizon in the training images, and a set of image segments $S_{below}$ that appear below the horizon in the training images. For every query image segment $s_i$, we first determine its position $p_i$ with respect to the horizon and then use only the corresponding set of training data to classify the segment.

Several recent approaches [12, 16] make use of a retrieval set of images, restricting training data to a set of images that are globally similar to the current query image. This approach can be understood as using separate retrieval sets for image segments that appear above, on, and below the horizon.

#### *Horizon Feature Concatenation*

The second method for incorporating horizon information into scene understanding is to combine color, texture, and horizon information into a single feature vector $f_i$. Because we have chosen histogram intersection as our metric when comparing feature vectors, we convert the position $p_i$ with respect to horizon into a 3-bin histogram with a single non-zero entry indicating either *above*, *on*, or *below*. Then we simply append this 3-bin histogram to the color and texture histograms described above to form the feature vector $f_i$. We individually normalize and weight each of the three histograms comprising $f_i$ to give equal weight to the color, texture, and horizon information when matching to other image segments.

#### *Horizon-Based Penalty Function*

Our final method penalizes otherwise similar image segments which happen to come from different positions with respect to the horizon. We first match a query segment against each labeled image segment in our database via histogram intersection $hist_{int}(f_i, f_j)$ based solely on color and texture. We then incorporate a penalty function $\phi(p_i, p_j)$ such that we discount the matching score between two image segments $s_i$ and $s_j$ which do not share the same position with respect to the horizon (i.e. $p_i \neq p_j$). We define the penalty function as follows:

$$\phi(p_i, p_j) = \begin{array}{|c|c|c|c|} \hline p_i \backslash p_j & above & on & below \\ \hline above & 0 & k_{small} & k_{large} \\ \hline on & k_{small} & 0 & k_{small} \\ \hline below & k_{large} & k_{small} & 0 \\ \hline \end{array}$$

Thus, we seek to maximize the similarity between each query image segment and all database image segments according to $similarity(s_i, s_j) = hist_{int}(f_i, f_j) - \phi(p_i, p_j)$. Note that the HBRS method can be viewed as a special case of this method with values $k_{small} = \infty$ and $k_{large} = \infty$, i.e. infinite penalty between segments which do not share the same position with respect to the horizon.

## RESULTS

We performed all image capture, ground truth annotation, feature extraction, and classification on a single mobile device: an HTC Sensation equipped with a 1.2 GHz dual-core CPU and 768 MB of memory, running the Android mobile operating system. Note that we do perform a one-time precomputation step offline on a desktop PC: determining the codebook of SURF features which serve as our texture descriptors as described above. We first extract SURF features from the SIFT Flow dataset [12] (2,688 images) and a subset of the Indoor Scene Recognition dataset of [13] (289 images), and then cluster these SURF features into 100 groups using k-means clustering. This is a one-time process and all other computation is performed on the smartphone.

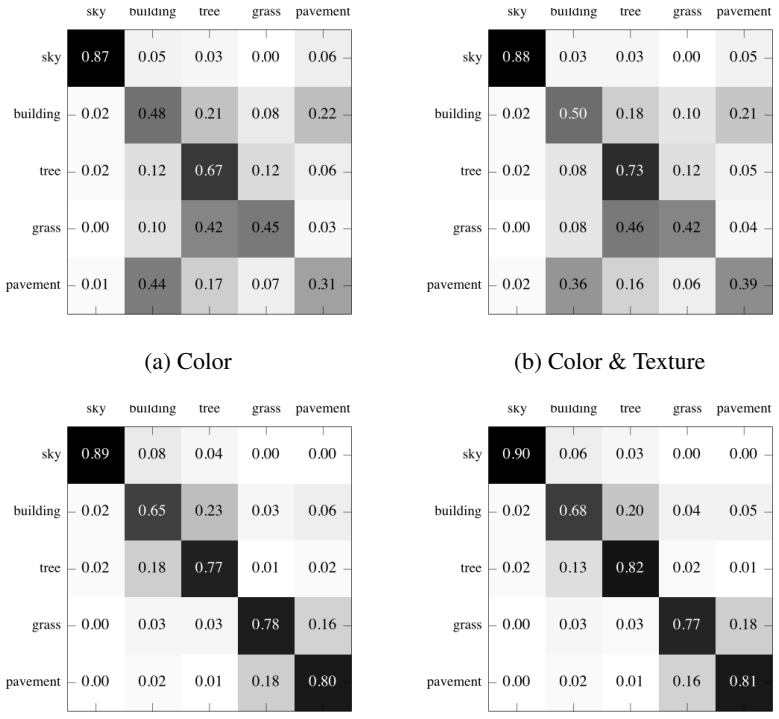
Our data set consists of 150 images captured and annotated on the smartphone, which we divide into 130 training images and 20 test images. All images were captured outdoors in an urban college campus setting and reflect a wide variety of locations, camera orientations, weather conditions, illumination conditions, and times of year. All images are downsized to 640 by 480 pixels before feature extraction. The 5 class labels we consider are *sky*, *tree*, *grass*, *building*, and *pavement*. The image segments we deal with are non-overlapping 32 x 32 pixel square patches. We use values of $k_{small} = 0.1$ and $k_{large} = 0.2$ for the Horizon-Based Penalty Function method. For all methods, we transfer the label of the single nearest neighbor database image segment to each query image segment to arrive at the final labeling result.

### Experiments

For each method described above, we label every image segment in all test images and compute the method's accuracy as the mean percentage of pixels correctly labeled. Results are summarized in Table 1 for both our Full Dataset (150 im-

| | Accuracy (Small Dataset) | Accuracy (Full Dataset) |
|---|---|---|
| Color | 54.5% | 50.5% |
| Color & Texture | 58.5% | 53.3% |
| Horizon-Based Retrieval Sets (HBRS) | **79.4%** | **77.4%** |
| Horizon Feature Concatenation | 79.4% | - |
| Horizon-Based Penalty Function | 79.1% | - |

Table 1: Comparison of Baseline & Horizon-Based Methods. For each method, we report accuracy as mean percentage of pixels correctly labeled across all test images. Horizon-based methods vastly outperform the baseline color and texture methods and show roughly equal performance, leading us to choose the more efficient HBRS method for subsequent experiments.




(a) Color

(b) Color & Texture

(c) Color & Horizon

(d) Color, Texture, & Horizon

Figure 5: Confusion Matrices for results based on (a) color alone, (b) color and texture, (c) color and horizon, and (d) color, texture, and horizon (HBRS). Inclusion of the horizon feature eliminates confusion between pairs of classes like building/pavement and grass/tree. The confusion that remains is almost entirely between scene elements that occur in the same physical locations in the world (e.g. grass and pavement are both on the ground).

| | Accuracy | Training Time per Image (seconds) |
|---|---|---|
| Gould et al. [2] | 67.1% | 94.8 |
| Our Approach (Color + Texture + Horizon) | 77.4% | 89.4 |
| **Our Approach (Color + Horizon)** | **75.5%** | **1.4** |

Table 2: Comparison to state of the art methods. The method of Gould et al. [2] is a state of the art scene-labeling method that runs on a desktop PC. In contrast our method (3rd row) runs on an Android smartphone, achieves greater accuracy (using the same testing and training sets), and is roughly 67 times faster (per image) at training time. Note that adding texture information (2nd row) improves our accuracy, but at significant time cost.

ages) and Small Dataset (a smaller subset of 54 images with 43 training and 11 testing images). Horizon-based methods, with up to 79.4% accuracy, clearly outperform the baseline color and texture methods which top out at 58.5%. It is interesting to note that all horizon-based approaches show roughly equal performance on the Small Dataset. This is actually welcome news, as the Horizon-Based Retrieval Sets (HBRS) method has the computational advantage that only a fraction of the total number of image segments must be matched against. Furthermore, note that the *only* difference between the baseline Color & Texture method and the HBRS method is that a restricted set of the data is searched. Thus, we get a roughly 21% to 24% boost in absolute accuracy while also speeding up the algorithm by searching a strictly smaller portion of the database. For all subsequent experiments, we therefore use the HBRS method exclusively (as well as the Full Dataset).

We also examine per-class accuracy, inter-class confusion, and how both measures improve with the addition of horizon information. The confusion matrices in Figure 5 show the effect of different feature combinations on per-class labeling accuracy and clearly demonstrate the advantage of our horizon-aware scene understanding approach. Using color alone, many instances of *grass* are labeled as *tree* and vice versa. Similarly, *building* and *pavement* are often confused. The addition of texture information helps only a small amount. However, the inclusion of horizon information via HBRS clears up the vast majority of this confusion, boosting the recognition rate of the *pavement* category from 31% to 81%, for example.

We show example labeling results for a variety of individual images in Figure 7. Across the board, when including horizon information we see marked improvements in labeling accuracy in a wide variety of situations: standard upright camera shots, images taken at highly irregular roll angles, and images in which the horizon is not even visible (either pointing up at the sky or down at the ground). It is even apparent from Figures 1a and 3 that the inertial sensors used to estimate the horizon are noisy, such that our horizon estimates are not always completely aligned with the image. In practice, this is not a problem, and we still see the benefits of horizon-based scene understanding even in the presence of noisy horizon estimates. For every test image, our horizon-aware method outperforms the baseline approach by a significant margin, ranging from 6% improvement up to 46% improvement (see Figure 7).

#### Comparison to State of the Art

We evaluate our system with respect to the state of the art scene labeling method [2] of Gould et al. This method runs on a Desktop PC and also attempts to estimate horizon information computationally, though only camera pitch is allowed to vary while roll is assumed to be zero. The method also requires a time-consuming batch training procedure which sees all the data at once, making it inappropriate for our usage scenario. Nevertheless, the results summarized in Table 2 show that our approach is able to achieve higher classification accuracy on our dataset while requiring roughly 1/67th as much time per training image (where both methods are trained and tested using our Full Dataset). This minimal training time is what allows our method to be interactive. Recall that our method is running on an Android smartphone, while the method of Gould et al. is running on a Desktop PC.

One reason that our method is able to achieve higher accuracy than [2] is that our database contains images with no restrictions on camera orientation, which violates several of the assumptions of [2]. Thus, more important than the absolute numerical results of this comparison is the fact that our method is able to deal with a more general set of images than existing methods, while simultaneously improving accuracy by incorporating horizon information from sensors on the device. (At the same time, note that [2] has been demonstrated to perform well on dozens of visual categories while our method has only been formally tested on the five categories in this paper. Larger scale evaluation will be an essential part of our future work.)

In designing our approach, we were forced to make tradeoffs between speed and accuracy in order to achieve interactive training and classification speeds. When we include texture information (the most accurate method in Tables 1 and 2), the running time of our system is dominated by dense SURF feature extraction, which takes roughly 88 seconds per image. However, the inclusion of horizon information actually makes texture less important, and by excluding texture our performance drops less than 2% (from 77.4% to 75.5%). For this small price in accuracy, we are able to both learn from labeled images and classify new images from the camera in roughly 1.4 seconds, making an interactive system feasible. This speed is due to our HBRS method, the relative simplicity of computing color histograms, and the use of a kd-tree for approximate nearest neighbor search among database image segments. We expect to gain further timing improvements by employing more efficient data structures and additional sensory information to guide this search.

#### User Study

Because our entire system runs on a mobile device, users of the system are able to measure the labeling performance while still out in the field collecting and annotating training images. In early experiments, we observed that the labeling accuracy over a set of test images improves as more training data are collected. This opens the door to active learning approaches in which the collection of new training data is directed by observation of errors made by the current system.

To understand the importance of this system feedback, we performed a small user study. Each of six non-expert users was instructed to capture and label outdoor images in order to train our system to learn five visual classes: sky, tree, grass, building, and pavement. When capturing a new image, half of the users were first shown the results of the system's current best estimated labeling of the classes in the scene. Only then were they allowed to label the images themselves. This *Active Learning* approach allowed the users to see what mistakes the system was making with

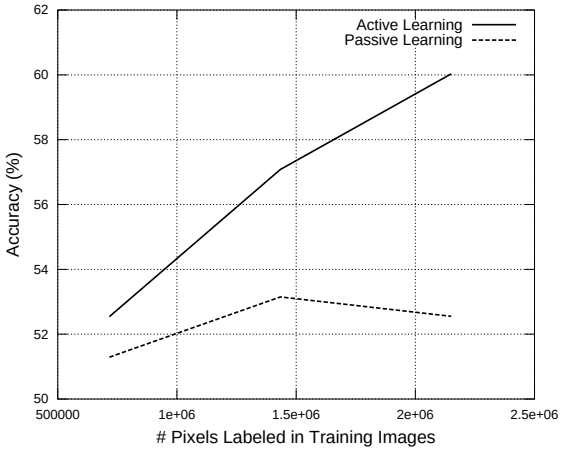



Figure 6: User Study: Active vs. Passing Learning. As users collect and annotate new images on a mobile device, performance generally improves over time. In our user study the Active Learning approach enables users to observe this change in performance while still out in the field collecting data, while the Passive Learning approach (like all existing approaches) does not give the user any feedback. As this plot shows, letting users see how the system is performing while they are still collecting and labeling data leads to improved classification accuracy for automated scene understanding.

its current training data. Since not every pixel in every image must be annotated, the user may choose to only annotate those parts of the image which the system did not already classify correctly. The other half of the users were never shown an automated labeling result by the system and got no feedback of any kind to know how effective their annotations were, an approach we call *Passive Learning*. Each user spent roughly 30 minutes collecting and labeling several dozen images each on an urban college campus. The slowest user collected and labeled 15 images, while the fastest user collected and labeled 28 images in this time frame.

The results of this study are shown in Figure 6, which shows average classification accuracy of the automated scene understanding system plotted against the number of pixels labeled by the user as they train the system. The results clearly show that the *Active Learning* approach greatly improves the rate at which the system's performance improves. We note that our system represents the first time that this approach has been feasible (for the present task) because it is the first to let the user see how the system is performing while still collecting training data out in the field. In this experiment, the same set of 20 testing images was used to evaluate the performance of the HBRS classification method across all users. These testing images were collected at a different time and place from the user study training images in order to demonstrate that our approach is able to generalize across data sets.

## CONCLUSION AND FUTURE WORK

We have introduced an orientation-aware scene understanding approach that improves labeling accuracy over current approaches, while simultaneously reducing computation time. This success is based on efficiently using features that are grounded in the physical world's coordinate system rather than in image coordinates. The entire system runs on a smartphone and can be trained from scratch out in the real world.

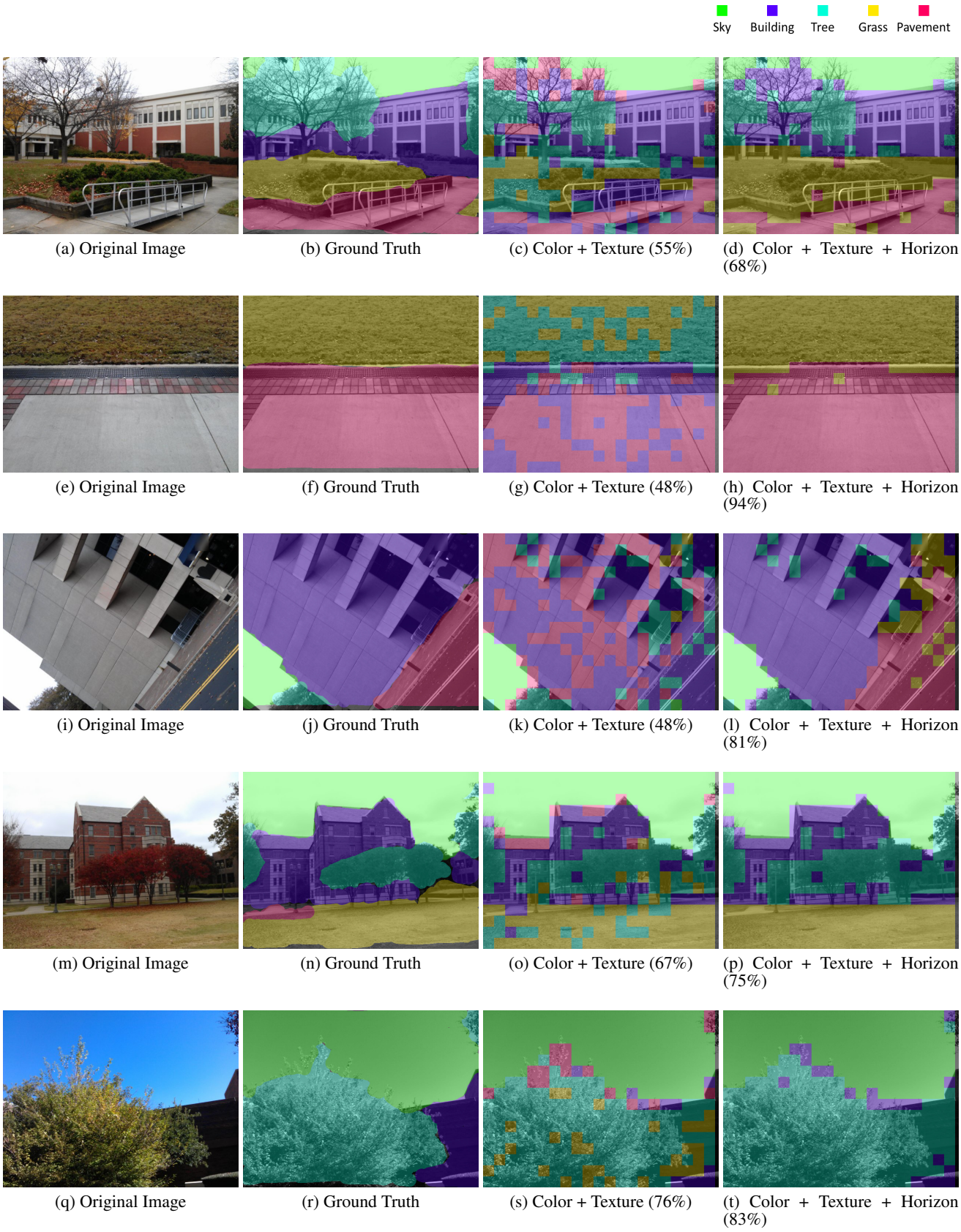
There are some limitations of our current approach. While we have removed most of the restrictions on camera orientation that previous approaches have imposed (upright camera, horizon visible within the field of view), we still assume, like most other approaches, that images are taken near the ground in order for the horizon to be a meaningful boundary. The additional sensors already embedded in smartphones provide promise that we can remove this restriction as well in future work, using either GPS or an altimeter to detect height above the ground. While our method shows some robustness to the presence of small hills and valleys (see Figure 1a), non-flat terrain is another challenge to all horizon-based methods which must be tackled in future work.

Another inherent limitation to the current data-driven approach is that as more training data is collected and labeled to improve accuracy, the size of the database increases, and therefore the algorithm has the potential to slow down. Memory and storage limitations may also become a problem. Neither of these issues has been a practical problem yet, but they remain important issues for scaling up the system. While tree-based data structures and hashing methods have the potential to keep the search for matching image segments fast, preventing the training data from exploding in size using a pruning strategy remains a problem for future work.

In future work, we also hope to integrate even more sources of sensory information about the camera and the environment into our scene understanding system. At the same time, we expect the increased speed of future mobile devices, combined with efficient new algorithms, to make our entire mobile scene understanding system run at video frame rates such that it becomes even easier to both teach our cameras about the world and to learn about the world from our cameras.

## REFERENCES

1. H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *ECCV*, pages 404–417. Springer, 2006.
2. S. Gould, R. Fulton, and D. Koller. Decomposing a scene into geometric and semantically consistent regions. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1–8. IEEE, 2009.
3. S. Gould, J. Rodgers, D. Cohen, G. Elidan, and D. Koller. Multi-class segmentation with relative location prior. *International Journal of Computer Vision*, 80(3):300–316, 2008.
4. D. Hoiem, A.A. Efros, and M. Hebert. Putting objects in perspective. In *CVPR*, volume 2, pages 2137–2144. IEEE, 2006.
5. D. Hoiem, A.A. Efros, and M. Hebert. Recovering surface layout from an image. *International Journal of Computer Vision*, 75(1):151–172, 2007.




(a) Original Image (b) Ground Truth (c) Color + Texture (55%) (d) Color + Texture + Horizon (68%)

(e) Original Image (f) Ground Truth (g) Color + Texture (48%) (h) Color + Texture + Horizon (94%)

(i) Original Image (j) Ground Truth (k) Color + Texture (48%) (l) Color + Texture + Horizon (81%)

(m) Original Image (n) Ground Truth (o) Color + Texture (67%) (p) Color + Texture + Horizon (75%)

(q) Original Image (r) Ground Truth (s) Color + Texture (76%) (t) Color + Texture + Horizon (83%)

Figure 7: Example Labeling Results: From left to right, the original images and ground truth scene labels, followed by baseline classification results (using color and texture), and our classification results (using color, texture, and horizon information). Reported numbers are for pixel-wise labeling accuracy compared to ground truth. In every case, our horizon-aware method outperforms the baseline approach by a significant margin, ranging from 7% improvement up to 46% improvement.

6. M. Hwangbo, J.S. Kim, and T. Kanade. Inertial-aided klt feature tracking for a moving camera. In *Intelligent Robots and Systems (IROS)*, pages 1909–1916, 2009.
7. M. Hwangbo, J.S. Kim, and T. Kanade. Gyro-aided feature tracking for a moving camera: fusion, auto-calibration and gpu implementation. *The International Journal of Robotics Research*, 2011.
8. S.H. Jung and C.J. Taylor. Camera trajectory estimation using inertial sensor measurements and structure from motion results. In *CVPR*, volume 2, pages II–732, 2001.
9. T. Kindberg, M. Spasojevic, R. Fleck, and A. Sellen. The ubiquitous camera: An in-depth study of camera phone use. *Pervasive Computing, IEEE*, 4(2):42–50, 2005.
10. D. Kurz and S. Ben Himane. Inertial sensor-aligned visual feature descriptors. In *CVPR*, pages 161–166. IEEE, 2011.
11. T. Lee and S. Soatto. Learning and matching multiscale template descriptors for real-time detection, localization and tracking. In *CVPR*, pages 1457–1464, 2011.
12. C. Liu, J. Yuen, and A. Torralba. Nonparametric scene parsing: Label transfer via dense scene alignment. In *CVPR*, pages 1972–1979. IEEE, 2009.
13. A. Quattoni and A. Torralba. Recognizing indoor scenes. In *CVPR*, 2009.
14. B.C. Russell, A. Torralba, K.P. Murphy, and W.T. Freeman. Labelme: a database and web-based tool for image annotation. *International Journal of Computer Vision*, 77(1):157–173, 2008.
15. J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. *ECCV 2006*, pages 1–15, 2006.
16. J. Tighe and S. Lazebnik. Superparsing: scalable nonparametric image parsing with superpixels. *ECCV 2010*, pages 352–365, 2010.
17. A. Torralba, K.P. Murphy, W.T. Freeman, and M.A. Rubin. Context-based vision system for place and object recognition. In *ICCV*, pages 273–280. IEEE, 2003.
18. S. You and U. Neumann. Fusion of vision and gyro tracking for robust augmented reality registration. In *Virtual Reality, 2001. Proceedings. IEEE*, pages 71–78. IEEE, 2001.
19. S. You, U. Neumann, and R. Azuma. Hybrid inertial and vision tracking for augmented reality registration. In *Virtual Reality*, pages 260–267, 1999.