האוניברסיטה העברית בירושלים
הפקולטה למתמטיקה ולמדעי הטבע
המכון למדעי המחשב

עבודת גמר לתואר מוסמך במדעי הטבע

Object Detection from Multiple Overlapping Views Using Semantic and Geometric Context in 3D

זיהוי אובייקטים מריבויי תמונות חופפות בעזרת קונטקסט גיאומטרי וסמנטי בתלת מימד

Avram Golbert
Advisor: Daphna Weinshall

1/1/2013

Object Detection from Multiple Overlapping Aerial Images

Thesis Avram Golbert
Advisor Daphna Wienshall
Hebrew University Computer Science
2012

Abstract

We present a method for object detection in a multi view 3D model. Out data is comprised of many overlapping aerial images with a 45 degree slant. These datasets are typical for the task of 3D reconstruction and include ground, roofs, building facades and many urban objects such as cars, plants and windows. These 3D models are becoming more and more common as 3D reconstruction algorithms improve, computation becomes cheaper and more and more uses are being found for them such as navigation, context for geo-referencing, urban planning etc. As the uses of these models grow beyond visualization so will the need for high level understanding of the scene and its contents. We use cars as a test case for a general framework for object detection in these datasets.

Typical aerial car detectors do significantly worse on this dataset because of the 45 degree angle photos that include many rectangular shaped objects, such as windows and doors, that are not visible in aerial datasets that are taken from the nadir. The good news is that the high redundancy that is necessary for the task of 3D reconstruction contains a lot of information that is generally not fully taken advantage of in classical object detector frameworks. In this work we focus on using all available information: highly overlapping views, geometric data and semantic surface classification in order to boost a simple 2D detector. Specifically, a 3D model is computed from the overlapping views, which represents the 3D scene and geometric relations. The model is segmented into semantic labels using height information, color and local planar features which describe the surfaces of the scene using high level classes such as roof, wall, road etc. A 2D detector is run on all

images separately and then detections are mapped into 3D via the model. The 3D space is used as a common language for all images and the detections are clustered in 3D and represented by 3D boxes. Finally, the detections, visibility maps and semantic labels are combined using a Support Vector Machine to achieve a more robust object detector.

We show that a simple car detector based on Haar-like features does not fare very well on these images, however integrating over multiple images, and using semantic and geometric context can achieve a high Precision-Recall ratio.

תקציר

במאמר הזה מוצגת שיטה לזיהוי אובייקט בעזרת מודל תלת מימדי וריבוי היבטים. הנתונים המוצגים מורכבים מריבוי תמונות אוויריות חופפות, שצולמו בזוית של 45° מתחת לאופק. נתונים (Datasets) מסוג זה אופייניים למשימות שיחזור תלת ממדיות וכוללות: קרקעות, גגות, חזיתות בניינים ואובייקטים עירוניים רבים כגון רכבים, צמחים וחלונות. מודלים תלת מימדים אלה הופכים להיות שכיחים יותר עם שיפור האלגוריתם של שיחזור תלת מימדי, הוזלת החישוביות, ומציאת יישומים נוספים כמו ניווט, הקשר להתוויות גיאוגרפיות, תכנון אורבאני וכדומה. ככל ששימושים במודלים אלה מתרחבים מעבר לשימוש לצורך הצגה חזותית פשוטה, כך עולה הצורך להבנה עילית של הזירה ומרכיביה. במחקר זה מבוצע שימוש ברכבים כמקרי ביקורת ליצירת מסגרת לאיתור אובייקטים במאגרי הנתונים.

גלאי רכב אוויריים טיפוסיים מצליחים פחות בצורה משמעותית מכיוון שתמונות בזווית של 45° מכילות אובייקטים מלבניים רבים כגון חלונות ודלתות, שאינם נצפים בתמונות אוויריות שצולמו מהנדיר. היתרון בשימוש במערכות נתונים מסוג זה, הוא שהיתירות הגבוהה הנדרשת לשחזור תלת מימדי מכילה מידע רב שלרוב אינו מנוצל במערכות סטנדרטיות לזיהוי רכבים. בעבודה זאת נעשה שימוש בכל המידע הזמין: מבטים בחפיפה גדולה, מידע גיאומטרי וקלסיפיקציה סמנטית על מנת לשפר את פעילותו של הגלאי הדו-מימדי. בפרט נעשה שימוש במודל תלת-מימדי משוחזר מתמונות חופפות המייצג את הסצנה התלת ממדית והיחסים הגיאומטריים. המודל מחולק לסגמנטים בעלי משמעות סמנטית תוך שימוש באינפורמציית גובה, צבע, מאפיינים מישוריים מקומיים אשר מתארים את הזירה בעזרת רכיבים בעלי משמעות כגון גגות, קירות, כבישים וכו'. בוצעה הרצה של הגלאי הדו-מימדי על כל אחת מהתמונות בנפרד ולאחר מכן הזיהויים מופו על ידי המודל למרחב תלת מימדי. המרחב התלת-מימדי משמש כשפה משותפת עבור התמונות. אם כך הזיהויים מקובצים בתלת מימד וכל מקבץ מיוצג על ידי תיבה תלת ממדית. לבסוף, הזיהויים, מפות ההסתרות וההקשר הסמנטי משולבים בעזרת Support Vector Machine ליצירת גלאי אובייקטים איכותי ואמין יותר.

תוצאות המחקר מראות שגלאי רכבים פשוט המבוסס על מאפיינים דמויי Haar משיג תוצאות דלות על מערכות נתונים אלה. יחד עם זאת שילוב של ריבוי תמונות ושימוש בהקשר גיאומטרי וסמנטי מניבים יחס גבוה של Precision-Recall.

Contents

1 Introduction

3D reconstruction is becoming more and more common as computing power increases and more methods are being developed. Standard graphics cards are now strong enough to generate photorealistic images of complex scenes in real-time. Typical data consists of multiple images with large overlap, where the camera's internal parameters and location are either known or estimated with SFM methods. The model is reconstructed using multiview geometry, and may be represented by such means as polygons, voxel space or planar disks; it typically contains no high level understanding or semantic interpretation. The widespread existence of such datasets and the existence of only low level understanding of the scene geometry have caused a need to build autonomous methods of extracting information from the data.

1.1 Object Detection in Images

Common methods for object detection in 2D images often use sliding windows of different sizes, where each window is tested for the existence of an object. A standard way of testing a given window is by extracting features and using machine learning to test it.

Viola and Jones [1] used Haar like features to describe a given patch. They define three kinds of features, two-three- or four rectangle features, see Fig. 1. The two rectangle feature defined as the sum of pixels in one rectangular area of the image minus the sum of pixels in an adjacent rectangular area, three rectangle feature is the sum of two rectangles minus the sum of a third rectangle and similarly the four rectangle feature is the sum of two rectangles minus the sum of another two rectangles. The size and location of rectangles in the given patch defines an overcomplete base. For a patch size of 24x24 the Haar like features define a set of 180,000 features.

Viola and Jones utilized a concept called Integral Images to enable fast computation of the Haar like features in all resolutions. The Integral images is defined as an image, ii, where the value of each pixel is the sum of all pixels in the upper left rectangle from the top left pixel to the given pixel; $ii(x,y) = \sum_{\substack{x'\leq x \\ y'<y}} i(x',y')$. The integral image can be computed quickly in $O(\#pixels)$ in one pass over the image since $ii(x,y) = ii(x-1,y) + ii(x,y-1) - ii(x-1,y-1) + i(x,y)$. Once ii is computed the sum of a given rectangle is $ii(bottom\ right) - ii(top\ roght) - ii(bottom\ left) + ii(top\ left)$. This representation allows us to compute the features at different resolutions without computing a pyramid of the original image as is often computed in other feature representations.

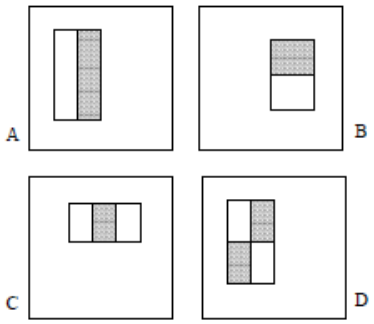


Figure 1: Types of Haar like features: two- three- four rectangle features. In all cases the sum of each rectangle can be computed by accessing 4 values of the integral image.

Viola and Jones used the given features to build a classifier using Adaboost. AdaBoost is a greedy algorithm that learns in every iteration a weak classifier $h_j(x)$ consisting of feature f_j , threshold θ_j and parity p_j indicating the direction of the inequality sign:

$$h_j(x) = \begin{cases} 1 & if\ p_j f_j(x) < p_j \theta_j \\ 0 & otherwise \end{cases}$$

Each weak classifier and its weight α_j are chosen to minimize the misclassification error for the current dataset with the current weights and the previous weak classifiers. Finally a classifier is defined:

$$H(\mathrm{x}) = \begin{cases} 1 & \sum_{\mathrm{t}} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{\mathrm{t}} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

A given patch x is detected as a car if $H(x) = 1$. Viola and Jones' main contribution, besides the choice of features and their fast computation, was to speed up the computation time of $H(x)$ by substituting $H(x)$ with a cascade of classifiers. Notice that the complexity, and also the success, of H is dependant on the number of features f used. They suggested to learn a cascade $H_1, H_2, \ldots, H_n$ such that the success of H_i on the patches that pass all previous classifiers will have a false negative rate smaller than ϵ and false positive rate smaller than δ. After n stages the overall false negative rate will be bounded from below by $(1-\epsilon)^{\mathrm{n}}$ and the false positive will be bounded from above by δ^{n}. Clearly this task gets more and more difficult and demands more and more features and the runtime of each stage will grow. However most patches that do not contain the desired object will fail in a very early stage, thus saving us a lot of unneeded computation in later stages. The result is a detector that works nearly as well in a fraction of the time.

Lienhart and Maydt [2] extended the set of features to include diagonal features and centered surround features while excluding the four-rectangle-features since these can be well represented by 2(e) and 2(g), see Fig 2. To support these new features they extended the integral image to incorporate a

possible rotation. In addition to the integral image of Viola and Jones they also calculated RSAT (Rotated Sum Area Table) for a rotation of 45 degrees. This paper also incorporated a contrast stretching of the form $\bar{\imath}(x,y) = \frac{i(x.y)-\mu}{c\sigma}$ $c \in R^{+}$. μ can easily be computed for a given area using RSAT, however σ needs information for the sum of sqruared pixels in the given area. To quickly compute this, another integral image of the squared pixels is needed. In total this paper uses 4 pre-computed images and boasts approximately 10% decrees in false alarms for a given hit rate in comparison to [1]

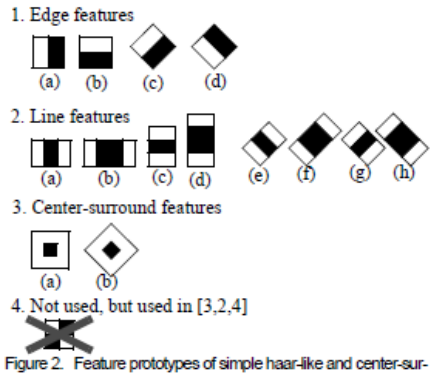


Figure 2. Feature prototypes of simple haar-like and center-surround features. Black areas have negative and white areas positive weights.

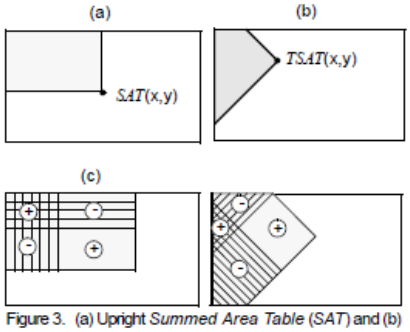


Figure 3. (a) Upright *Summed Area Table* (*SAT*) and (b) *Rotated Summed Area Table* (*RSAT*); calculation scheme of the pixel sum of upright (c) and rotated (d) rectangles.

Figure 2 extended set of Haar like features and the rotated integral image (RSAT) for 45 degree rotated features

Although [1-2] developed and tested their methods on databases for face detection there is nothing inherent in the algorithm for face detections. H. Grabner et al [3] learned a car detector for large aerial images using extended Haar features along with local orientation histogram and local binary patterns. They trained the classifier using online boosting which drastically reduces the need for a large training set. In each learning iteration the user marked one wrongly labeled detection and the classifiers were adjusted. In this way marking multiple examples that aren't informative is avoided and only significant examples are labeled. The online learning is done using online AdaBoost. In the offline AdaBoost all training samples are available and the weak classifiers $h_1, h_2, \dots h_n$ are train sequentially on all the training samples with their current weight and then the samples reweighted before the next learning stage. Online boosting introduces a new concept "Selectors" which holds a set of M weak classifiers and selects one of them $h_{sel}(x) = h_m^{weak}(x)$ according to an optimization criterion (specifically the estimated error of each weak classifier). When a new sample is introduced each selector trains all its weak classifiers and then best one is selected, the weight of the new sample is updated and passed to the next selector to continue the training. In the online boosting algorithm a strong classifier is available at all times.

Kluckner et all [4] extend the idea of online learning by using 3D information to automate the labeling in each step in order to detect and label false

positives. The database they used is the UltraCamD camera from Microsoft Photogrammetry which consists of 11500 × 7500 pixel images taken from the Nadir with high overlap. The images are taken by flying in strips with a 60% overlap between strips and an inner strip overlap of 80%. They used the multiple views to assign a depth for each pixel. The reference image is compared with 2 other overlapping images where each depth defines a pixel matching to the other images and the pixels are compared using a similarity function, in this case Normalized Cross Correlation (NCC). Since the images are very close to each other, using only triangulation will result in erratic and large changes in the depth map caused by noise in the pixel matching, occlusions and changes in lighting. To overcome the inherent noise a global optimization scheme is used to achieve a more consistent solution. Intuitively we expect neighboring pixels with similar appearance to have similar depth values. Instead of performing global optimization over all image pixels, which is computationally expensive, they introduced super pixels and local planar constraint. The image was segmented into super pixels and each super pixel was assumed to correspond to some 3D plane. Neighboring super pixels should ideally reside on the same plane. The problem was formalized as a Markov Random Field (MRF) combining a Data term for assigning a given segment to a certain plane, and a Smoothness term for assigning neighboring segments to different planes. The Data cost is the sum of pixel correspondence defined by the homography induced by the 3D plane. The smoothness cost is a discontinuity penalty that incorporates the common border lengths and the mean color similarity. Loopy Belief Propagation was used to find an approximate optimal assignment of planes to each segment.

Figure 3 a small part of the depth image with the corresponding color image and detected cars. The depth map maximizes both the correlation to matching pixels in other images and global depth consistency between neighboring pixels

This extra data was used to learn an aerial car detector. Like Grabner et al [3] they used online Adaboost starting from a single positive example, and after each iteration the depth image was used to automatically label the false positives. A simple method was used to identify false positives. They assumed that for true car detections all pixels in the current window (and a small area surrounding it) should all have similar depth values. All pixels in the detection window were fitted to a plane, and if this plane had a slant above a certain threshold the detection was declared to be false and was reweighted for the next iteration of online learning. Since it's not possible to automatically label true positives using the height map, only initially hand labeled positives were available during the learning. To compensate, when a

hand labeled car was not detected, it was reweighted and no false positives were added in that round. It is important to emphasize that once the detector was learned, no depth information is calculated or used when evaluating a new image. This can be seen as an advantage and a disadvantage. On the one hand it can run on a single image with no extra demands. On the other hand when multiple images, or depth information are available, it doesn't utilize this valuable information to improve detections.

1.2 Using Context to Boost Detection

While methods for patch based detection and recognition are making progress using better descriptors and learning methods, others are seeking to use more information than just the pixels in a given patch. This includes a variety of context and scene understanding cues, in order to boost performance for object detection. Oliva and Torralba [5] review current research about the importance of context in human vision. Real world object co-occur with objects of the same type as well as different types of objects. A computer screen will often be above a keyboard and a desk, and a fire hydrant will be above a grey-street layer with a dark building type background above it, see Fig. 4. Experiments show that humans find objects with higher percentage when the object is in its natural context in multiple levels: semantic, spatial and pose. Semantic: a table and chair are probably present in the same images, whereas an elephant and a bed are not. Spatial configuration: a keyboard is expected to be below a monitor. Pose: chairs are oriented towards the table, a pen should have a particular pose relative to the paper to be useful for writing and a car will be oriented along the driving directions of a street. In typical experiments the subjects are shown an image or a few consecutive images for less than 200ms, and asked to identify the objects and/or scene. These experiments show the importance of the location. A plate should be on table but also could be on wall or cabinet, a fire hydrant will always be on top of the sidewalk, not below the ground plane or floating in the air. Object recognition was shown to be more accurate if the relationship between the context and the object is strong, and the observers' accuracy was facilitated if the target (e.g. a loaf of bread) was presented after an appropriate scene (e.g. a kitchen counter) and impaired if the scene–object pairing was inappropriate (e.g. a kitchen counter and bass drum)

Figure 4 images averaged over the LabelMe dataset. The images were translated and scaled to put the object in the center before averaged. It's easy to see that the context of these objects.

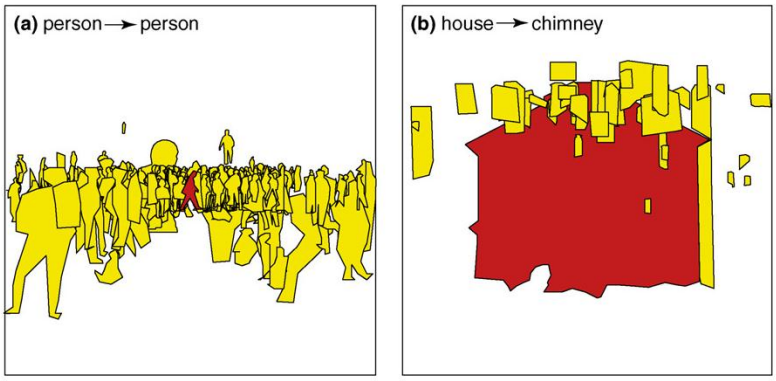


Figure 5: locations were objects may be found in relation to the center object (shown in red). In the left image we see people in relation to a person in the center. In the right image we see likely locations of a chimney in relation to a house.

Heitz et al. used Spatial Context by using the co-dependence between local scenes and the presence of certain objects to label the scene and boost object detection simultaneously. They worked on two very different datasets: PASCAL VOC 2005/2006 and satellite images from Google Earth. The PASCAL dataset was used to perform detection of cars, bicycles and motorbikes in street scenes and detection of cows and sheep in rural scenes. The Google Satellite images were used to perform car detection. The image was segmented into super pixels and a local scene feature was extracted for each segment. N implicit local scene classes were learned from the training images, which can intuitively be identified as classes such as street, building, trees, water, etc. The relationships, R, between the N classes and the objects were also learned using a Bayesian network model and Expectation-Maximization. During testing the base detector is run over the image and all windows, W, above a minimum threshold are considered. Finally, given the features F_i for each window, an assignment T(the presence of objects) was estimated that maximizes $p(T|F,R,W)$.

The probability of an assignment T is a summation of all possible assignments, S, of each segment to a class

$$p(T|F,R,W) = \sum_{S} p(T,S|F,R,W)$$

Since all possible assignments, S, is exponential in the number of classes, it's not feasible to search over all possible assignments for S,T and choose the maximum likelihood. Instead, they used the Gibbs sampling which is an iterative method. An initial assignment for T,S are chosen and are alternately updated by re-sampling with the distribution $P(T|S,F,R,W)$ and $P(S|T,F,R,W)$. Since the Bayesian model assumes the S_i labels are independent given T and T_i are independent given S

$$P(S|T,R,F,W) = \prod P(S_i|T,R,F,W)$$

$$P(T|S,R,F,W) = \prod P(T_i|S,R,F,W)$$

which can be calculated easily by further decomposing:

$$P(S_i|T,R,F,W) \propto P(S_i) \cdot P(F_i|S_i) \cdot \prod_{j,k} P(R_{ijk}|T_j,S_i)$$

After the last iteration T is used as an estimate of the maximum likelihood of the object detections.

Hoiem et al [7] used geometric context, photometric context and local objects in a naive Bayes model. The task combined detection of pedestrians and cars in images taken from street level. Geometric context attempts to recognize surface orientation, ground, vertical and sky, with 5 subclasses of vertical: planar facing "left", "center", and "right", and non-planar "solid" and "porous". The image was segmented into super pixels and features containing color, texture, shape, location and 3D geometry (long lines, intersections of lines, parallel lines and texture gradients) were extracted. A classifier was trained using Adaboost with weak learners based on eight-node decision trees.

Photometric context is an estimation of viewpoint. All images were assumed to be taken from approximately human height facing the horizon. Given the location of the horizon and the height of the camera, the expected size of an object in the image can be calculated depending on its location in the image. They used a simple Gaussian model prior for camera height and orientation, which were also assumed independent. The scene was modeled by a graphical model: geometric surfaces are independent given their corresponding object identities and object identities are independent given the viewpoint, see Fig. 6. These assumptions make the decomposition task possible and each variable is dependent on only one other allowing these dependences were learned. Given an image the maximum likelihood assignment was found using the Pearl's belief propagation algorithm.

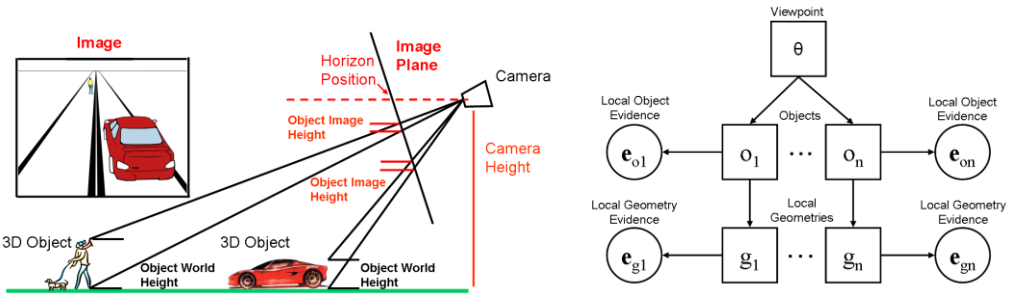


Figure 6: pose estimation. Given the camera height and location of horizon object sizes are governed by their location. On the right is the graphical model.

Divvala et al. [24] performed an extensive experiment testing the importance of different types of context. They combined local pixel context, 2D scene gist, 3D geometric, semantic, geographic and photogrammetric context. Local pixel context is simply using a slightly larger bounding box that includes pixels from the surrounding area. In this way we implicitly expect green pixels in close proximity of "cow" and road in close proximity of "cars". 2D scene gist includes global image statistics to classify the gist of the scene. 3D scene includes physical layout of the scene, support surface, surface orientations, occlusions, etc. Semantic context includes scene category, objects present in the scene and types of surfaces. Geographic context is a novel context they presented and includes the actual location the image was taken. This information includes terrain type, land category, elevation, population density etc. Photogrammetric context includes image orientation, height, internal camera parameters, radial distortion, etc. All these types of context can directly affect the probability of the presence, location and size of different types of objects in the image. A classifier is learned for each type of context. For each detection object presence, object location and object size estimates are calculated using each of the defined context classifiers. A logistic regression classifier is trained using the above features on the VOC'08 validation set. Their experiments showed the importance of reasoning about an object within the context of the scene, as the average precision of the original detector was boosted from 18.2 to 22.0.

1.3 Integration of 3D information

The use of 3D scene information to enhance object detection has been made even more explicit. It's common in robotics to find systems that combine standard images with auxiliary input such as laser sensors, Sonar, Radar as well as navigational data such as GPS and inertial systems. These inputs can be combined to generate a full map of the locations of the platform over time as well as a 3D map of the scene. This task is called Simultaneous Localization and Mapping (SLAM), see section 2.1.

Posner et al [8] used a system comprised of an RGB camera, Lidar and inertial measurement system for the task of adding semantic labeling to the map. Typically, pixel features are extracted from the image for the task of learning a semantic classifier. Posner et al incorporated the 3D information as well. 3D laser points were accumulated over a time window of length Δt into

the past. Thus, a 3D point cloud was assembled which represents the 3D scene in a limited time around t. The 3D points were segmented into planes following a divide-and conquer approach: a given point cloud was discretized into cubic cells and planes were fitted locally using RANSAC. Plane segments, for which the support (i.e. the number of inliers) was less than a threshold, were discarded. Planes obtained in neighboring cells were merged according to two constraints relating to relative surface orientation and translation. The merging criteria for neighboring planes demands that the normal of the planes be closer than a maximum angle and the distance between them, defined by the distance of the center of mass projected onto the other plane, be smaller than a given distance. Finally these planes were projected back onto the image to define a depth and normal for each pixel as well as a geometric segmentation. Appearance features were extracted from the image for both color and texture. Color features were represented by a 15 bin histogram for Hue and Saturation of pixels in a 15x15 neighborhood as well as variance. Since the platform is always on the ground facing the horizon (as in [7]) the spatial location of the pixel is very informative as spatial context, thus the x,y location was also added to the feature. An all-vs-one classifier was learned for each class using Gaussian Kernel SVM, and each pixel was labeled with the classification that gave the greatest margin.

This work did not use any spatial or temporal inner-pixel reasoning, i.e neighboring pixels were classified independently and a 3D point projected onto 2 different images may be labeled differently in each. This was partially addressed by voting over each planar segment and showed better results but was only expanded on in [9], also Posner et al. In this work the images were segmented both by the 3D planes as well as standard image super pixel techniques. Instead of using SVM to classify each pixel, a probabilistic framework was defined and each segment was assigned a probability for each class. More specifically, a feature is computed for each pixel in the segment, and the feature is associated with a "word" from a vocabulary learned during the learning phase. The segment is assigned a binary vector $(z_1, \dots, z_{|V|})$ where $|V|$ is the size of the vocabulary and $z_i = 1$ if the word v_i is present in the segment, 0 otherwise. The probability of class k given an observation z, $P(C_k|z)$ is proportionate to $P(z|C_k)$ and some priors. Calculting $P(z|C_k)$ is dificult to learn unless approximating that all z_i are independent given Ck; $P(z|C_k) = \prod P(z_i|C_k)$. This however is a bad assumption as different words are very dependent on each other. Instead the full dependence is modeled by constructing a dependence tree. i.e $P(Z_p|z, C_k) \cong P(Z_p|Z_{pq}, C_k)$ where pq is the parent of q. The Chau Liu tree is the optimal tree approximation. Once each segment in the image has a probability for each class, an optimal assignment is one that maximizes both the inner segment probability and a temporal probability. More explicitly we expect neighboring segments to have the same label with high probability. The problem was formulated as a Markov Random Field using $1 - P(C_k|z_s)$ as the data cost of assigning label C_k to segment s, and the smooth cost is a simple probability of two different classes being neighbors which can easily be measured from the training set.

Posner et al added temporal context in [10] by adding links in the MRF over N images (in their experiments $N = 3$). Segments from different images were

linked if their corresponding 3D points overlap by more than 20%. The MRF was solved for segments in all N images.

Kluckner et al. [11] used a similar concept for the task of semantic labeling in aerial images combining laser scans with color images although their method was different. Unlike Posner et al. an initial segmentation of the image was not performed. Instead, features were extracted per pixel, and class probability was estimated per pixel using Random Forests. A Conditional Random Field was defined on the four neighborhood graph of the image pixels where edges between neighboring pixels are weighted according to the height difference in the Lidar images. The energy is minimized using graph cuts based on linear programming.

Figure 7: Aerial Semantic labeling. Shown the original image and depth map from Lidar.

Figure 8: associated Semantic labeling by performing Graph Cuts on 4 neighborhood image grid

Bischof et al. [13] used similar data for the task of street layer extraction and car detection. Overlapping images were used to generate depth maps, and objects that are specific to roads were detected and used as starting points for full street layer segmentation. They employed a specifically designed Gabor filter to detect zebra crossing (cross walks). Again, a 4 neighborhood grid was built for the image and the edges between pixels were weighted according to

color and height similarity. Since the images were taken from the nadir, the street layer was assumed to be well connected in the image. Zebra crossings from multiple images were projected onto the reference image using the 3D information and Dijkstra's shortest path algorithm was used to find lowest cost paths between zebra crossings. Zebra crossings with large distances were filtered out as false detections, while the remaining zebra crossings and the paths between them were assumed to be points on the street layer. A Thin Plate Spline (TPS) was fitted to these points using Least Squares as well as the color distribution for streets. All other points in the image that match the TPS and color distribution were labeled as street layer. Cars were detected using Adaboost, although any other method is possible, and detections that do not lie on the street surface were removed. Although they used overlapping images, the process was done independently for each image followed by interpolation obtained by projecting the street layer onto the Digital Terrain Map (DTM).

Our approach takes this research direction a bit further starting from the reconstruction of a 3D scene model from the overlapping views. Inference is done in the 3D space using all images equally as multiple measurements simultaneously and not defining one image as a reference while using a few overlapping images as support like [8-13]. We detect static objects in the model by using detections from all images and 3D semantic labeling simultaneously. More specifically, the camera location and orientation are calculated for each image using Slam [14] and then a dense 3D model is calculated [15-17]. A sliding window detector is run on each image at 6 different rotations and each image detection is translated to a 3D Bounding Box using the camera calibration and 3D model. All 3D Bounding Boxes are clustered into a smaller set of representative 3D Bounding Boxes. This allows us to infer from many images while overcoming obstructions and greatly varying viewpoints. A multiclass semantic labeling of the model is performed using geometric information, local planes and color information from all images. We show that using multiple overlapping viewpoints and context greatly improves the initial performance of the 2D detector.

2 Our Method: Object Detection from Multiple Views

Our method is described in Algorithm 1 below. The input is a set of overlapping images $\{\boldsymbol{I_i}\}$. We first reconstruct the 3D scene from these images in order to obtain estimated location and orientation for each image and a 3D model – a mesh (V, T) consisting of vertices and triangles, see Section 2.1. Next, we look for objects in the images - cars defined by location, orientation and size $d = (l, o, s)$. In Section 2.2 we describe how for each image I_i a set of cars d_{ij} is detected and assigned weights w_{ij} using a cascade of weak classifiers. In Section 2.3 we describe how each detection d_{ij} in image I_i is mapped to a 3D Bounding Box D_{ij}, and how a 3D Bounding Box is projected

onto an image.

Algorithm 1: object detection in multiple overlapping views

Input: set of images $\{\boldsymbol{I_i}\}$.
Output: set of object detections in images

1. (V, T) ← Reconstruct the 3D scene and obtain 3D model.
2. For each image I_i
   - $\{d_{ij}, w_{ij}\}$ ←Run sliding window detector in 6 rotations
   - $\{D_{ij}, W_{ij}\}$ ←Map 2D detections into 3D
3. $\{D_k, W_k\}$ ←Cluster 3D detections and choose representatives $\{D_k\} \subset \{D_{ij}\}$ and corresponding weights $\{W_k\}$.
4. Compute semantic labels for each vertex in the model $v \in V$
   - $\{f_v\}$ ←compute feature vector containing 3D and photometric information
   - $\{W_v^s\} \in [0,1]_{|V|\times|S|}$ ← compute a semantic weight for every semantic class
5. Classification with Semantic Context
   a. $\{f_{D_k}\}$ ← compute feature vector containing semantic information from all vertices in the neighborhood of D_k, local information and detection weight W_k
   b. Final car classification using SVM

The detections d_{ij} are mapped into 3D in order to achieve common parameterization and used together to infer information about the existence of objects. They are first clustered into a set of 3D boxes D_k. The local information around each D_k is used along with the 2D information to better understand the scene and improve the detection. In Section 2.4 we define semantic classes S: *Ground, Wall, Vegetation, Roof, Tiled Roof and Car*. Every 3D vertex $v \in V$ is assigned a semantic weight, W_v^s, for each semantic class. In Section 2.5 the semantic weights and previously calculated information in the neighborhood of D_k are used to construct a semantic vector $f_{D_k}^S$; final classification is obtained with SVM.

2.1 3D Reconstruction

It's common in robotics to find systems that combine standard images from multiple viewpoints with auxiliary input such as laser sensors, Sonar, Radar as well as navigational data such as GPS and inertial systems. These inputs can be combined to generate a full map of the locations of the platform over time as well as a 3D map of the scene. This task is called Simultaneous Localization and Mapping (SLAM). Usually the auxiliary information isn't complete or is not exact enough and multiple overlapping data from different times must be combined to produce a coherent map. Visual Slam, or

Monocular Slam, uses only a single moving camera and also measurements estimating its location and orientation if available.

Denote:

- T^t, Ψ^t - The camera parameters (location and orientation) at time t
- R^t – The rotation matrix for the camera at time t as defined by Ψ^t
- K - The camera's intrinsic matrix. Used to project a 3D point to a pixel in homogeneous coordinates. Assumed to be the same for all cameras.
- P_i - The 3D location of point i.
- p_i^t - The 2D pixel representing the 3D point i on the camera at time t.

  The projection function is: $p_i^t = \begin{bmatrix} \frac{v_i^t(0)}{v_i^t(2)} \\ \frac{v_i^t(1)}{v_i^t(2)} \end{bmatrix}_{2x1}$; $v_i^t = KR^t(P_i - T^t)$

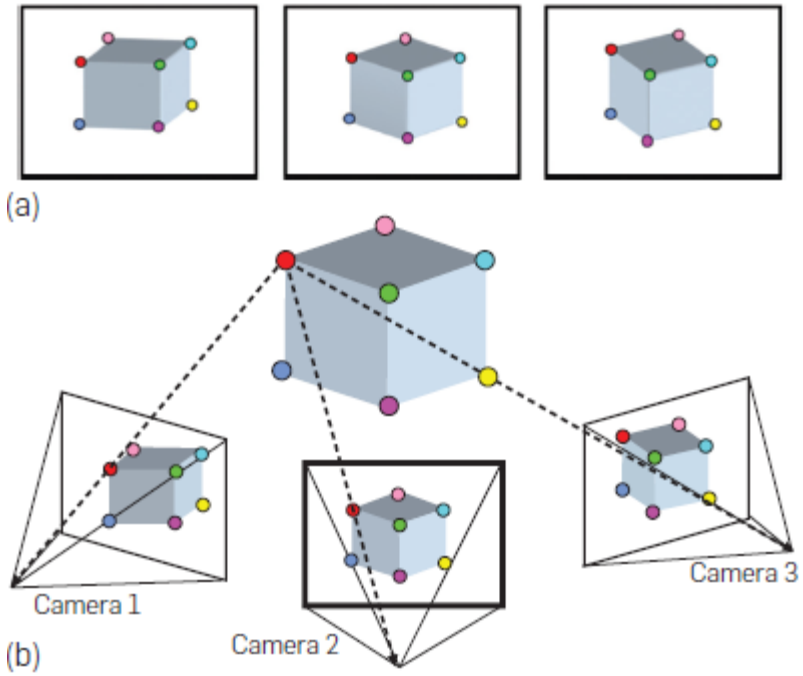


Figure 9: 3 images of the scene. 3D points are tracked through all images. A 3D Point, P_i is projected onto image t at p_i^t

The first step is to generate corresponding points, also known as tracks, between the images. A track, $\{\tilde{p}_i^{t_1}, \tilde{p}_i^{t_2}, \dots, \tilde{p}_i^{t_n}\}$, is a list of pixels in different images that correspond to a single point in the 3D word. Given this set of tracks and an initial guess for $\tilde{T}^t, \tilde{\Psi}^t$, the problem can be formalized as finding the correct orientation of these images and the 3D locations of the points that were extracted using maximum likelihood estimation.

We assume that the given measurements have some Gaussian error with known covariance:

- $\tilde{p}_i^t = p_i^t + n_i^t$ where $n_i^t \in \mathcal{N}(0, \Delta_{\mathrm{i}}^{\mathrm{t}})$ and p_i^t is the true projection of P_i on image t
- $\tilde{T}^t = T^t + n_T^t$ Where $n_T^t \in \mathcal{N}(0, \Delta_{\mathrm{T}}^{\mathrm{t}})$ and T^t is the true camera position at time t
- $\tilde{\Psi}^t = \Psi^t + n_\Psi^t$ Where $n_\Psi^t \in \mathcal{N}(0, \Delta_\Psi^t)$ and Ψ^t is the true camera orientation at time t

We see that given the normal distribution, the probability of a measurement $\tilde{p}_i^t$ given its true projection p_i^t is a normally distributed variable:

$$P(\tilde{p}_i^t | p_i^t) = P(\tilde{p}_i^t - p_i^t) = P(n_i^t) = \frac{|\Delta_{\mathrm{i}}^{\mathrm{t}}|^{-\frac{1}{2}}}{2\pi} e^{(\tilde{p}_i^t - p_i^t)^{\mathrm{T}} (\Delta_{\mathrm{i}}^{\mathrm{t}})^{-1} (\tilde{p}_i^t - p_i^t)}$$

The problem is then formalized as finding the parameters P_i, T^t, Ψ^t that maximize the likelihood of the measurements $\tilde{p}_i^t, \tilde{T}^t, \tilde{\Psi}^t$:

$$argmax_{P_i, T^t, \Psi^t} \left\{ \prod_{i,t} P(n_i^t = \tilde{p}_i^t - p_i^t) \cdot \prod_{\mathrm{t}} P\left(n_T^t = \tilde{T}^t - T^t\right) \cdot \prod_{\mathrm{t}} P\left(n_\Psi^t = \tilde{\Psi}^t - \Psi^t\right) \right\}$$

$$= argmax_{P_i, T^t, \Psi^t} \left\{ \prod_{i,t} \frac{|\Delta_{\mathrm{i}}^{\mathrm{t}}|^{-\frac{1}{2}}}{2\pi} e^{(\tilde{p}_i^t - p_i^t)^{\mathrm{T}} (\Delta_{\mathrm{i}}^{\mathrm{t}})^{-1} (\tilde{p}_i^t - p_i^t)} \cdot \prod_{i} \frac{|\Delta_{\mathrm{T}}^{\mathrm{t}}|^{-\frac{1}{2}}}{2\pi} e^{(\tilde{T}^t - T^t)^{\mathrm{T}} (\Delta_{\mathrm{T}}^{\mathrm{t}})^{-1} (\tilde{T}^t - T^t)} \cdot \prod_{\mathrm{t}} \frac{|\Delta_{\Psi}^{\mathrm{t}}|^{-\frac{1}{2}}}{2\pi} e^{(\tilde{\Psi}^t - \Psi^t)^{\mathrm{T}} (\Delta_{\Psi}^{\mathrm{t}})^{-1} (\tilde{\Psi}^t - \Psi^t)} \right\}$$

$$= argmax_{P_i, T^t, \Psi^t} \left\{ \sum_{i,t} (\tilde{p}_i^t - p_i^t)^{\mathrm{T}} (\Delta_{\mathrm{i}}^{\mathrm{t}})^{-1} (\tilde{p}_i^t - p_i^t) + \sum_{t} \left(\tilde{T}^t - T^t\right)^{\mathrm{T}} (\Delta_{\mathrm{T}}^{\mathrm{t}})^{-1} \left(\tilde{T}^t - T^t\right) + \sum_{\mathrm{t}} \left(\tilde{\Psi}^t - \Psi^t\right)^{\mathrm{T}} (\Delta_{\Psi}^{\mathrm{t}})^{-1} \left(\tilde{\Psi}^t - \Psi^t\right) \right\}$$

The above cost function corresponds to the weighted Sum of Squared Error (SSE) when we use the inverse of the covariance matrices $(\Delta_{\mathrm{i}}^{\mathrm{t}})^{-1}, (\Delta_{\mathrm{T}}^{\mathrm{t}})^{-1}, (\Delta_{\Psi}^{\mathrm{t}})^{-1}$ as the weights.

Once the camera calibration is known, dense reconstruction algorithms

attempt to "fill in the blanks" of the sparse 3D map. There are a few ways to represent the scene, most notably Voxels (Volumetric Pixels), Range Maps and Polygon Mesh. Voxels form a regular 3D grid where each voxel can encode a binary variable, occupied/solid vs. vacant, or as a function encoding the distance to the surface. This representation is useful for its simplicity and its quick query time for a given location in space. However, its cost in memory consumption may become impractical for large areas since the volume of the scene grows polynomialy in space and resolution. Range maps are a set of 2D images for known locations and encode the distance to the surface. Range maps avoid the need to resample the 3D space and offer an easy way to break certain tasks into smaller ones that can be easily managed and computed in parallel. Polygon meshes represent a surface as a set of connected planar facets. They are efficient to store and render and are therefore a popular output format for multi-view algorithms.

Given a point P on the scene surface and two or more cameras with known orientation in which P is visible, we can demand the appearance of the projected pixels be similar. This is called the Photo consistency of the scene. The photo consistency measure varies between methods and the similarity measure between pixels is generally not intrinsic to the reconstruction algorithm. The Photo consistency function can be divided into two types, Scene Space or Image Space. The former integrates over the scene and evaluates the accuracy of the model by comparing the pixels in all relevant images. An example of such a measure can be the variance of the pixels. The latter uses the scene to warp, or project, one image onto another, and then compare the reprojected image to the original, called the Prediction Error or Reprojection Error. An example of the comparison method can be Normalized Cross Correlation (NCC), Sum of Square Differences (SSD), etc. Scene Space error functions are integrated over the surface of the scene and thus often tend to prefer smaller surfaces, whereas prediction error is integrated over the set of images of a scene and thus ascribe more weight to parts of the scene that appear frequently or occupy a large image area.

Goesele et al. [14] presented a simple yet robust method to build a depth map for each image and merge them into a mesh. In the first stage a set of range maps are built, one for each image. A depth is assigned per pixel encoding the estimated distance to the surface of the scene from the camera center. Each image is compared to a set of target images, V, that have similar viewpoints. The target images are chosen using the tracking information from the SLAM with the assumption that images with many mutual tracks are similar enough to be successfully compared at the pixel level.

Two cameras with images I_1, I_2 and known orientations R_1, R_2 define a fundamental Matrix $F \in R_{3\times3}$ such that any 3D point P projected to pixels $p_1, p_2 \in R_{3\times1}$on images I_1, I_2 respectively fulfill $p_1^T F p_2 = 0$. Alternatively, a given pixel p_1 in I_1 defines a line $l_{p_1} = p_1^T F$ in I_2. This line is called the epipolar line and every point on the epipolar line in I_2 uniquely defines a depth for p_1, see Fig. 10.

Every pixel p in the reference camera and every depth, d, is assigned a grade according to the similarity of the projection on the target images. The

Normalized Cross Correlation between pixel p and view j, $NCC(p, V_j, d)$ is computed between an $m \times m$ window centered around p and the corresponding windows centered around the projections in each of the views R_j. If two views show the same surface we expect to see a high NCC score for some value of d. If, in contrast, there is an occlusion or other compounding factor, the NCC value will typically be low for all depths. We wish to rely on a depth value only if the window in the reference view correlates well with the corresponding window in multiple views. We therefore define that a depth value is valid if $NCC(p, V_j, d) > thresh$ for at least two views in V. The set of all views with NCC larger than thresh for a given depth d is denoted as $C_V(d)$. For a valid depth d we compute a correlation value as the mean of the NCC values of all views in $C_V(d)$:

$$corr(d) = \frac{\sum_{C_V(d)} NCC(p, V_j, d)}{\|C_V(d)\|}$$

For each pixel the depth is chosen to be the value that maximizes $corr(d)$ or none if no valid depth is found. This method is very simple and easy to implement and has a short runtime on modern multi-threaded platforms since we don't assume any dependence between pixels in the range map. Using only values that are greater than a minimum thresh implicitly handles obstructions caused by the 3D nature of the scene and the viewpoint changes. The Normalized Cross Correlation method normalizes each patch according to its mean and STD and is therefore robust to light changes. However, it is not rotation invariant. Therefore, for databases with few images, Goesele's method results in large areas with low confidence or no depth. For databases with many images such as the Dino and Temple dataset, with over 300 images each, the method shows good results with completeness of 98% and 80% respectively, but only 86% and 57% respectively when using only 50 images.

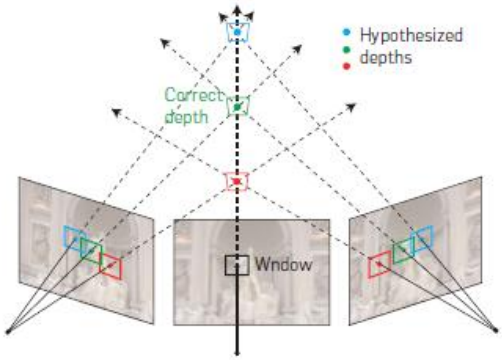


Figure 10: Given a pixel and an image window around it, a number of depths along its viewing ray are hypothesized. At each depth, the window is projected onto the other images. At the true depth (highlighted in green), the consistency score is at its maximum.

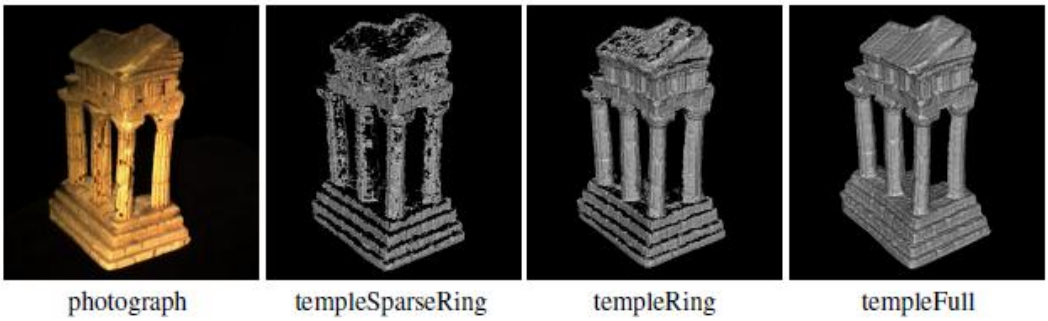


Figure 11: Dense recursion results for temple data using 16, 47 and 317 images.

The depth maps are merged into a single surface mesh using the volumetric method of Curless and Levoy [17]. The space is discretized into voxels and an implicit continuous function $D(x)$ is estimated. The function represented is the weighted signed distance of each 3D point, x, to the nearest surface. Each point x in the voxel space is projected onto all range maps $I_1 \dots I_n$ and distances $d_1(x), \dots, d_n(x)$ are calculated using the distance between x and camera location and the estimated depth in the range map. Weights $w_1(x), \dots, w_n(x)$ are assigned using the correlation grade calculated by the NCC described above. Then

$$D(x) = \frac{\sum_n w_i(x) \cdot d_i(x)}{\sum_n w_i(x)}$$

Finally an iso-surface is extracted using marching cubes at points $x\ s.t\ D(x) = 0$

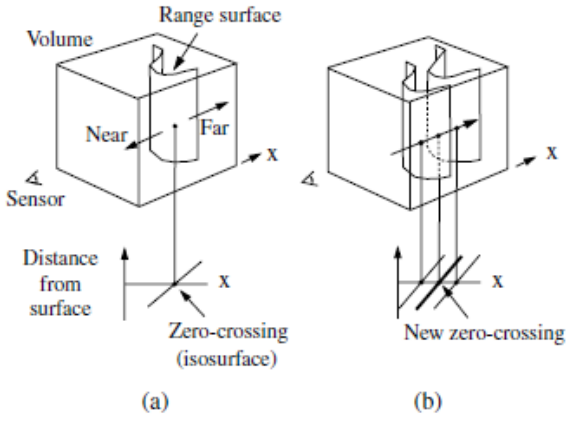


Figure 12: (a) A range sensor looking down the x-axis observes a range image. Following one line of sight down the x-axis, we can generate a signed distance function as shown. The zero crossing of this function is a point on the range surface. (b) The range sensor repeats the measurement, but noise in the range sensing process results in a slightly different range surface. Following the same line of sight as before, we obtain another signed distance function. By summing these functions, we arrive at a cumulative function with a new zero crossing positioned midway between the original range measurements.

2.2 2D Detections

We train a single 2D detector for each oriented object. We first train a cascade of 22 weak classifiers using Haar-like features [1,2], retrieving those windows that pass all levels of the cascade. The cascade is run on 6 different rotations of the image, see Fig. 13. For angle $\theta \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ the image is rotated and the detector is used to find detections that are parallel to the image axis. Detection d_{ij} in location (x, y) in a rotated image is translated to location $R(\theta)^t \begin{bmatrix} x \\ y \end{bmatrix}$ with orientation $-\theta$ in the original image.

As is customary, we perform post processing to overcome multiple detections of the same object in close locations. In order to take location, size and orientation into account, we define two detections to be "close" if all corresponding corners are no more than $\alpha \cdot size$ apart. In our experiments we set $\alpha = 0.3$. All detections are clustered using disjoint sets, i.e. two sets A, B will be joined if they contain items that are close. Finally, we set $w_{ij} = \#\{detection\ in\ cluser\}$.

Figure 13: example of 2D detections in 3 rotaions of the original image

2.3 Clustering in 3D

Each 3D object is visible in many images. The 2D detector may fail to detect it in some images but succeed in others. In order to collect data from different images and use them together, the detections are mapped into 3D. Using the calculated camera locations, a vertex $v \in R^3$ is projected onto the image plane with the projection matrix P_i and, conversely, a pixel in image I_i defines a ray that intersects the 3D Mesh at point p. Thus detection d_{ij} in image I_i is projected to a 3D box D_{ij} by projecting the center c_{ij} of detection d_{ij} onto the mesh at point C_{ij}, which is used as the center point of D_{ij}. The 3D orientation is calculated by projecting the 2D orientation onto the x, y plane $(z = 0)$ around C_{ij}. D_{ij} is always assumed to be parallel to the x, y plane, and is assigned weight $W_{ij} = w_{ij}$.

Next, we seek a subset of all 3D boxes from all images that best explains the 2D detections. We define a measure of similarity between two 3D boxes

$$s\left(D_{i_1 j_1}, D_{i_2 j_2}\right) = Vol(D_{i_1 j_1} \cap D_{i_2 j_2}) / Vol(D_{i_1 j_1} \cup D_{i_2 j_2})$$

This can easily be calculated since the detections are parallel to the x, y plane and can be calculated as the area of the intersection and union of 2D rectangles. The z dimension is ignored when there is an overlap, whereas $s\left(D_{i_1 j_1}, D_{i_2 j_2}\right) = 0$ otherwise, to help prevent the similarity function from diminishing too quickly.

We represent this problem as a multi-labeling problem. The goal is to choose a representative for each D_{ij} that best describes it, requiring that similar detections have the same representatives. We define a graph $G = (V, E)$ whose nodes are the 3D detections, and edges are drawn between any intersecting detections: $V = \{D_{ij}\}, E = \{(D_{i_1 j_1}, D_{i_2 j_2}) | s(D_{i_1 j_1}, D_{i_2 j_2}) > 0\}$. A labeling of the graph is a function $L: \{D_{ij}\} \rightarrow \{D_{ij}\}$; it is a mapping from each 3D detection to a representing 3D box. We use Graph Cuts to minimize the total energy [21-23].

$$E(G, L) = \sum_{D_{ij}} W_{ij} \cdot \left(1 - s\left(L(D_{ij}), D_{ij}\right)\right) + \sum_{\substack{(D_{i_1 j_1}, D_{i_2 j_2}) \in E, \\ L(D_{i_1 j_1}) \neq L(D_{i_2 j_2})}} \min\left(W_{i_1 j_1}, W_{i_2 j_2}\right) \cdot s\left(D_{i_1 j_1}, D_{i_2 j_2}\right)$$

The first sum is the data fidelity term - the cost of assigning D_{ij} to $L(D_{ij})$. The second sum is the smoothness term - the cost of assigning neighboring detection to different labels. We use the image of the labeling function as the representative set: $\{D_k\} \leftarrow \{D_{ij} | \exists i' j', L(D_{i' j'}) = D_{ij}\}$, and each label's weight is assigned the sum of weights over the cluster

$$W_k = \sum_{W_{D_{ij}} \in L^{-1}(D_k)} W_{D_{ij}}$$

2.3.1 Clustering Implementation Details

In our experiments described below, mapping all 2D detections into 3D resulted in approximately 80k detections. Building the entire graph G, and running Graph Cuts on all possible labeling, would have been prohibitive despite recent improvements in the runtime of Min cut-Max flow and Graph Cuts algorithms. Graph Cut works by running iterations of min-cut on intermediate graphs G_α for every potential label α. The runtime is controlled by the size of G_α and the number of labels. To improve runtime we divide the problem into smaller problems by splitting G into sub graphs and run Graph Cut on each independently. We split G using disjoint sets with an aggressive distance threshold $\alpha = 0.4$, i.e. two disjoint sets $A, B \subset \{D_k\}$ will be joined if $\max_{a \in A, b \in B} s(a, b) > \alpha$. This is done quickly by utilizing a KD-tree

architecture to search for close neighbors. An aggressive threshold can be used since we expect the detections around a car to be dense and separating them is unlikely. In our experiments this resulted in small graphs that contained at most a couple thousand detections.

To further improve runtime we consider only a small portion of the potential labels. For each D_k we add the four best representatives $argmin_{L(D_k)}\{Data(D_k)\}$. Alpha expansion is performed only on these labels, where during construction of the intermediate graph G_α we restrict the graph to nodes $\{D_k | s(D_k, \alpha) > 0\}$. This greatly reduces runtime and guarantees that the representative $L(D_k)$ of D_k is not too far from D_k.

2.4 Semantic Labeling

The mesh is represented by a set of 3D vertices (point cloud) and a set of polygons, but it doesn't contain any high level information. It's not possible to ask questions such as "What's in this area?", "Is there a lot of vegetation here?", or "Is there a car here?". Intuitively we expect to find certain objects in a natural context - bears in the forest, toaster in the kitchen, etc. Here we represent the context by learning semantic classes: *Ground, Vegetation, Wall, Roof, Tiled Roof and Car.* We use information from the surrounding geometry as well as information from the images that view each vertex $v \in V$ to assign weights $W_v^s \in [0,1]$ for every semantic class. Local geometric information is achieved by adapting the methods of [19] to 3D models, as opposed to range maps, in order to segment the model into planar and non-planar areas.

We use these planar segments when creating a feature vector f_v as follows:

1. Vertex Normal: average normal of polygons that contain this vertex
2. Vertex Height: height above Digital Terrain Map (DTM); when DTM is not available, RANSAC is used to obtain one
3. Planar Segment Normal: the normal of the segment
4. Planar Segment Type (binary): planar or non-planar
5. Planar Segment size: the size of the segment
6. RGB histogram, 15 bins
7. RGB standard deviation
8. Hue histogram, 15 bins

For each semantic class a "One Vs. All" classifier was learned using AdaBoost; it is a greedy algorithm that learns in every iteration a weak classifier $h_j(x)$ consisting of a feature f_j , a threshold θ_j and a parity p_j indicating the direction of the inequality sign:

$$h_j(x) = \begin{cases} 1 & \text{if } p_j f_j(x) < p_j \theta_j \\ 0 & \text{otherwise} \end{cases}$$

Each weak classifier and its weight α_j are chosen to minimize the misclassification error for the current dataset with the current weights and the previous weak classifiers. Finally, a strong classifier is defined:

$$h(x) = \begin{cases} 1 & \sum_t \alpha_t h_t(x) \geq \frac{1}{2} \sum_t \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

Alternatively, we can define $\alpha_j' = \frac{\alpha_j}{\sum_t \alpha_t}$ and $h'(x) = \sum_t \alpha_t' h_t(x)$. In this way $h'(x) \in [0,1]$ and $h'(x) \geq \frac{1}{2}$ iff $h(x) = 1$. We use this slight modification to be able to compare the confidence of the different classifiers by setting $W_v^s \leftarrow h_s'(v)$. In Fig 14 each vertex, v, was colored with the corresponding color of $argmax_s(W_v^s)$.

2.5 Boosting Classification with Semantic Context

When examining a potential detection of a car, it's important to consider its surroundings and not only what this location looks like when projected onto different images. Thus, we may expect the vertices that constitute a car and its surroundings to have high scores in the *Road* and *Car* coordinates of W_v^s, and low scores in its *Roof* coordinate.

Specifically, each 3D detection D has 3D volume that intersects the model and contains N_D vertices. The scores for all classes are averaged over all vertices in D to create a six coordinate vector $f_D^s = \frac{1}{N_D} \sum_{v \in D} W_v^s$. This vector is concatenated to the 3D cluster weight W_D and $\frac{W_D}{|\{I_{vis(D)}\}|}$, where $\{I_{vis(D)}\}$ is the group of images in which D is visible. The last coordinate is important since it normalizes the grade against areas with higher visibility, such as roofs that naturally get a higher grade than obscured areas like cars in alleyways. However, it is not sufficient since it will over-represent a single false detection in an otherwise occluded area.

In summary, each 3D detection D is represented by the vector f_D^s, which contains information from all images, the surrounding context and its visibility.

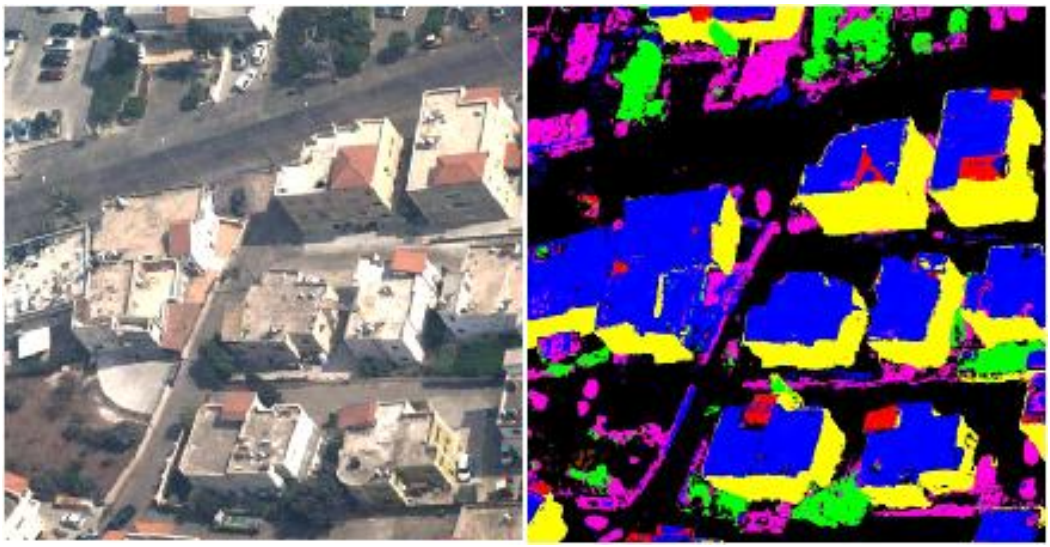


Figure 14 : Original image and semantic labels. Classes are roof, tiled roof, wall, ground, vegetation and car in blue, red, yellow, black, green and pink respectively

2.5.1 Classification Implementation Details

Here we use the feature vector f_D^s to learn an SVM classifier with RBF kernel $k(x_i, x_j) = e^{-\gamma\|x_i - x_j\|^2}$, which distinguishes cars from background.

To aid classification, we uniformly scaled the data to [0,1], since $w^s \in [0,1]$ while in some cases $W_D > 10^3$. We used a small part of the data and cross validation, in order to find good values for the penalty parameter of the error term C and the kernel parameter γ. Specifically, we used grid search on exponentially growing values of $\gamma = 2^{-4} \dots 2^4$ and $C = 2^{-4} \dots 2^{10}$ and chose the pair that maximized the Area Under Curve (AUC) of the PR. Simple percent correct wouldn't work on this unbalanced dataset, where a trivial negative classifier achieves accuracy of 96%. We chose the pair $(C, \gamma) = (0.125,256)$, although there was a wide range of pairs that performed almost equally well. SVM was then trained using v-fold cross-validation with 10 groups while weighting the positive instances by 25 to overcome the unbalanced dataset.

3 Experiments

3.1 Data and 3D Reconstruction

We used a set of 420 aerial images that together cover a ground area of $400 \times 400m$ with a surface resolution of $15cm/pix$. The images were taken at a $45 - 60°$ below the horizon. The images have a large overlap and every ground point is visible in at least $80 - 100$ images (unless hidden by other objects). The images came with GPS and Inertial Navigation System (INS) measurements. They were corrected using Slam techniques and a dense 3D model was reconstructed. Stationary cars were reconstructed as well and are visible in the 3D model.

3.2 Training

All the data, images and model were split into two groups according to the ground area they had covered. A geographic area covering a quarter of the entire area, and all data associated with it, was designated as training data; the remaining data was used for testing. This division, according to ground coverage rather than choosing pictures at random, guaranteed that we won't have different instances of the same car in both the train and test sets.

We hand labeled the location and orientation of the cars in the 3D model and projected them onto the images to create 2D training and test sets. Each 2D instance was rotated so that the wheels are aligned with the x axis of the image and only fully visible instances were used for training (Fig. 15). Negative examples were selected randomly from areas of the image that don't have any vehicles, even partially visible (Fig. 16).

Figure 15 : Example of multiple views with large baseline and various levels of visibility. Images were rotated to align with the x axis and only fully visible examples are used for training (marked in red)

Figure 16 : A few negative examples

We built a cascade of weak classifiers. Each weak classifier must have a false positive rate smaller than λ and a detection rate no lower than $1-\epsilon$. Thus a cascade of N weak classifiers achieves a false hit rate lower than λ^N and detection rate higher than $(1-\epsilon)^N$ on the training set. We set $\lambda = 0.5$, $\epsilon = 0.01, N = 22$.

To train the semantic classifiers we hand labeled vertices in the part of the model that corresponds to the ground area designated for training. Classifier h'_s for class $s \in S$ was trained using all vertices with label s as positive examples and vertices with different labels as negative examples. The AdaBoost algorithm described in section 2.4 was used with 20 iterations. These results are shown in Fig. 14 where each vertex v was assigned the label corresponding to $argmax_{s \in S}\{h'_s(v)\}$. We can see for example that *Vegetation* is confused with *Ground* and with *Car* but almost never with *Roof*. *Tiled Roof* may be mistaken for *Roof* or *Wall*.

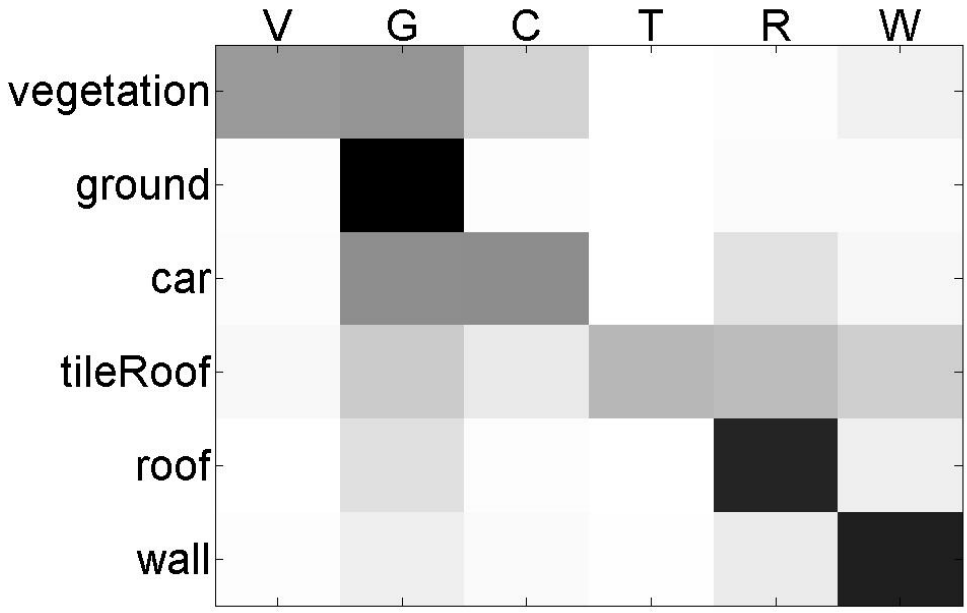


Figure 17 : Confusion matrix for semantic labels. On the left is the true label and on the top is the detected label. Black denotes high detection rates.

3.3 Results

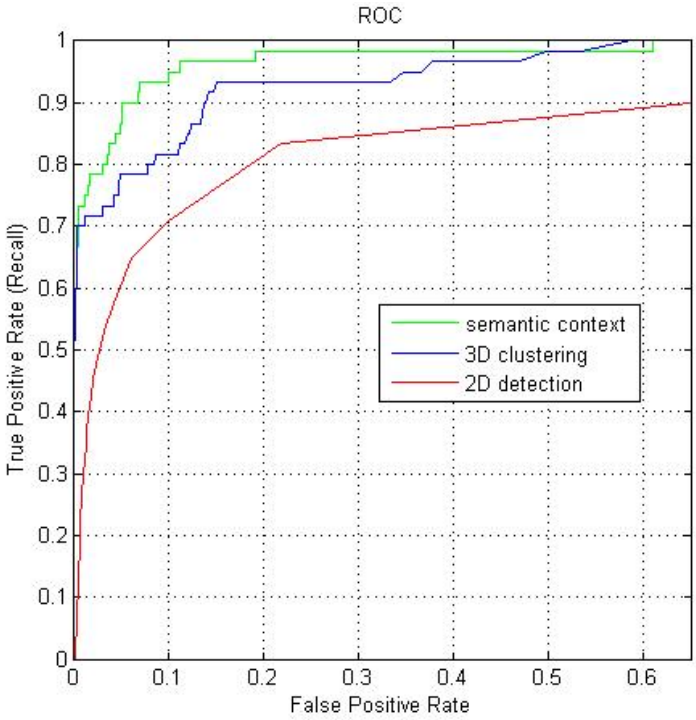
We show results comparing a pure 2D car detector vs. our method both with and without the use of context. It's important to emphasize that our method detects objects in 3D, while the alternative sliding window method detects 2D instances in single images. This means that our method detects each car only once, while the 2D method detects all instances of the same car in different images as unrelated detections. Note that this set is very unbalanced with only 60 cars in the entire set. This means that only 1:20000 of the model vertices are cars. This is greatly visible in the Precision Recall (PR) curves, especially in the 2D curve. For this reason we show both the Receiver Operating Character (ROC) and the PR. The main difference in these two curves is that in PR the false positives are normalized against the true positives, while ROC is normalized against the true negatives. For this reason the ROC is much more "forgiving" of false positives when the data is unbalanced [20].

2D Detector: The 2D detector is very noisy and never achieves $Precision > 0.2$, where chance detection would achieve $Precision = 0.04$ on this dataset. This can be explained by the angles in which the images were taken. Many walls are visible with box like windows. Haar-like features measure gain changes in rectangular areas, and therefore many false positives occur on corner/boxy objects such as windows and solar panels. We tried using an alternative detector [6] that was trained on Google Satellite images from the nadir with similar pixel resolution, but its performance was not better than chance.

Our method with 3D Clustering: After clustering we were left with 1500 detections, most with small weights that could be discarded without losing any true positives, see Fig. 18. We achieve $Recall = 1, FPR > 0.65$.

Our method with Semantic Context: For each detection D found by the clustering stage we constructed the feature vector f_D^S as described in Section 2.4. Each detection contained some 2-3 thousand vertices.

Fig. 18 shows the performance of the 3 methods: 2D detector, our method with 3D clustering, and our method enhanced by semantic context. We see that clustering detections from multiple images achieves 50% recall with no false positives. At higher Recall values, using Semantic Context can reduce false positives by a factor of 2 or more. This is hardly visible in the ROC curve in this unbalanced data. In the PR curve a reduction in false positives is very noticeable since it's proportionate to the true positives.



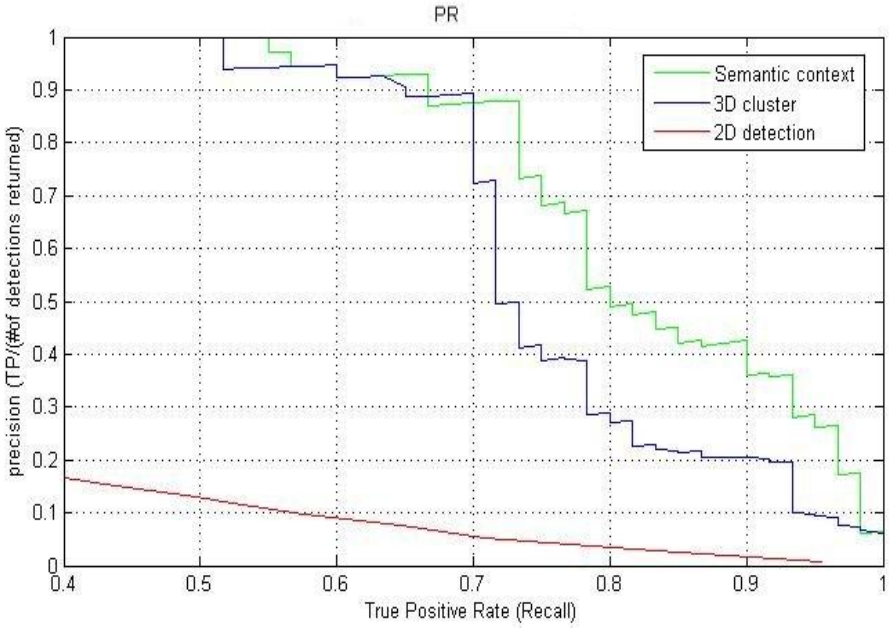


Figure 18 : ROC and Precision Recall Curve. For Recall = 0.5 both 3D clustering and Semantic Context have perfect precision. For recall above 0.7 Semantic Context improves precision by a factor of 2.

4 Summary and Discussion

We have described a method for finding static objects in multiple overlapping views using 3D reconstruction. Our method combines detections from a 2D sliding window detector. The detections from all images are translated into 3D where they are all considered together. Since true detections are stable over multiple images, clustering in 3D results in more reliable detections. Using the semantic labels of the 3D model as context and using visibility of each object location in the images further improves the detections.

Our choice for the 2D detector was motivated by the desire to use a generic, simple 2D detector for 2 reasons. The first was to show that our framework is not dependent on a designated car detector [3,4]. Clearly a better car detector will simplify the task assigned to our framework. It is possible that a highly designated car detector will have far less false positives to sift through in consequent stages of our framework, in which case, perhaps, a simpler method of clustering will suffice. This brings us to our second motivation for using a simple detector. Our framework is not designed for a specific object and we wanted a detector that is not specific for any one object type. The same framework with the same detector can be trained to detect any object that has a statistical correlation with its geometric and semantic surroundings.

In order to combine detections from different images we used the 3D representation as a common language. Once all the 2D detections were converted into a 3D Bounding Box we were able to compare them, measure the distance between them and the extent to which they differ. The similarity measure between two detections should not only measure the location but also the size and orientations. If the detection were represented as a vector of location, size and orientation a simple distance measure could be tricky. We chose to use the ratio between the intersection and union as a similarity function since this is a compact way to calculate and represent all of the above. The similarity can be assumed to be zero if the centers are far enough from each other. This approximation will prove important to avoid a full graph in the clustering stage.

During clustering we use Graph Cuts. This method was chosen, instead of standard clustering methods such as k-means and mean-shift, for several reasons. First and foremost, Graph Cuts offers control over the representation of the constraints. By not assigning edges to the graph we can explicitly restrict the set of possible solutions. Since we have some insight into the type of errors inherent in 2D detectors, we can assign different weights according to that probability. In our case, by restricting the graph edges to detections with nonzero intersections, we place an explicit limit to the size of a cluster. In this dataset, with many overlapping images, an algorithm, such as k-means, mean-shift or hierarchal clustering, is likely to result in large clusters. Since each cluster represents a single detection, it's clear we don’t want a cluster to grow too big.

Second, by choosing only the most promising links, and placing edges only between “promising” detections we can explicitly lower the number of edges

in the graph to $O(N)$ which is a huge reduction in computation time when solving the Graph Cuts.

Future work may explore a clustering method that uses a statistical model to representation of the links between potential detection. We can study the what kind of errors a 2D detection generates around an object, in space scale and rotation and how this error is projected into 3D. We tested a Gaussian Model Mixture (GMM) clustering scheme, however, using Gaussian models on rotation is problematic, and the runtime on a cluster this size was also unfeasible. It is possible to use a Gaussian Model to convert a cluster into a maximum likelihood detection instead of the average of the cluster as we described.

Once a few key detection were selected to represent all the detections we wished to use all the information at hand for each cluster to answer the simple question "Is this cluster indeed a car?". Each cluster has information from every image that saw it (after taking obstructions into account), the weight assigned to it (zero if there is no detection there) and the semantic information from its surroundings. Intuitively we expect a car to have little "roof", a lot of "street", and a lot of "car" in it. These labels were summed into a vector of probabilities that can be interpreted as exactly that, the amount of "roofness" or "carness" in this area. Using a leave-one-out method we could learn a separator between "car" and "background" using SVM with RBF Kernel. We found that the performance is robust to very a large range of assignments to the parameter of the error term C and the kernel parameter γ. The kernel parameter could take a wide range of pairs with almost identical classification results.

The choice of SVM for the inference stage is not an obvious one. Future work can focus on an MRF model that connects different image observations, the geometric and semantic context and possibly even other detections in this area. Such models are popular when using context to boost classification. They typically neglect inner model dependencies such as detection-detection and goemtric-goemtric dependencies in the presence of a detection. These approximations allow a simple decomposition of the statistic model into smaller independent expressions that can estimated either from the data, or as a prior.

5 References

[1] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In Proc. of IEEE Conference on Computer Vision and Pattern Recognition, Kauai, HI, December 2001.

[2] Rainer Lienhart and Jochen Maydt. An Extended Set of Haarlike Features for Rapid Object Detection. IEEE ICIP 2002, Vol. 1, pp. 900-903, Sep. 2002.

[3] H. Grabner et al., "On-Line Boosting-Based Car Detection from Aerial Images," *ISPRS J. Photogrammetry and Remote Sensing,* vol. 63, no. 3, 2010, pp. 382-396.

[4] S. Kluckner, G. Pacher, H. Grabner, H. Bischof, and J. Bauer, "A 3D teacher for car detection in aerial images," in Proc. IEEE Int. Conf. Comput. Vis., 2007, pp. 1–8.

[5] Oliva, A., Torralba, A.: The role of context in object recognition. Trends CognSci (2007)

[6] G. Heitz and D. Koller. Learning spatial context: Using stuff to find things. In Proc. ECCV, 2008.

[7] D. Hoiem, A. A. Efros, and M. Hebert. Putting objects in perspective. IJCV, 80(1), 2008.

[8] Posner, I., Schroeter, D., Newman, P.M.: Describing composite urban workspaces. In: ICRA. (2007)

[9] Posner, I., Cummins, M., & Newman, P. (2008, June). Fast probabilistic labeling of city maps. In Proceedings of Robotics: Science and Systems IV, Zurich, Switzerland.

[10] Posner, I., Cummins, M., Newman, P., "A generative framework for fast urban labeling using spatial and temporal context" Autonomous Robots, 26(2-3):153-170. April 2009.

[11] B. Douillard, D. Fox, and F. Ramos. A spatio-temporal probabilistic model for multi-sensor multi-class object recognition. In Proc. of the International Symposium of Robotics Research (ISRR), 2007.

[12] S. Kluckner et al., "Semantic Classification in Aerial Imagery by Integrating Appearance and Height Information," *Proc. ACCV2009,* 2009.

[13] Georg Pacher, Stefan Kluckner, and Horst Bischof, "An Improved Car Detection using Street Layer Extraction" Moravske Toplice, Slovenia, Computer Vision Winter Workshop 2008

[14] B. Triggs, P. McLauchlan, H. R.I, and A. Fitzgibbon. Bundle adjustment - a modern synthesis. In Vision Algorithms'99, pages 298–372, 1999.

[15]Seitz, S., Curless, B., Diebel, J., Scharstein, D., Szeliski, R.: A comparison and evaluation of multi-view stereo reconstruction algorithms. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition. (2006)

[16]Goesele, M., Curless, B., Seitz, S.: Multi-view stereo revisited. In: Proc.

IEEE Conf. on Computer Vision and Pattern Recognition. (2006)

[17] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In SIGGRAPH, pages 303.312, 1996.

[18] A Practical Guide to Support Vector Classification. Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin Department of Computer Science National Taiwan University, Taipei 106, Taiwan

[19] D. Gallup, J. Frahm, and M. Pollefeys, “Piecewise planar and nonplanar stereo for urban scene reconstruction,” in Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on. IEEE, 2010, pp. 1418–1425.

[20] J. Davis and M. Goadrich. The relationship between precision-recall and roc curves. Technical report #1551, University of Wisconsin Madison, January 2006.

[21] Efficient Approximate Energy Minimization via Graph Cuts. Y. Boykov, O. Veksler, R.Zabih. IEEE TPAMI, 20(12):1222-1239, Nov 2001.

[22] What Energy Functions can be Minimized via Graph Cuts? V. Kolmogorov, R.Zabih. IEEE TPAMI, 26(2):147-159, Feb 2004.

[23] An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision. Y. Boykov, V. Kolmogorov. IEEE TPAMI, 26(9):1124-1137, Sep 2004.

[24] S. K. Divvala, D. Hoiem, J. H. Hays, A. A. Efros, and M. Hebert. An empirical study of context in object detection. In CVPR, 2009.

[25] L. Zebedin, A. Klaus, B. Gruber-Geymayer, and K. Karner. Towards 3d map generation from digital aerial images. International Journal of Photogrammetry and Remote Sensing, 60:413–427, Sept. 2006.