Classification and segmentation of orbital space based objects against terrestrial distractors for the purpose of finding holes in Shape from Motion 3D reconstruction

T. Nathan Mundhenk[1], Arturo Flores, Heiko Hoffmann
HRL Laboratories LLC, 3011 Malibu Canyon Rd, Malibu, CA, USA 90265

ABSTRACT

3D reconstruction of objects via Shape from Motion (SFM) has made great strides recently. Utilizing images from a variety of poses, objects can be reconstructed in 3D without knowing *a priori* the camera pose. These feature points can then be bundled together to create large scale scene reconstructions automatically. A shortcoming of current methods of SFM reconstruction is in dealing with specular or flat low feature surfaces. The inability of SFM to handle these places creates holes in a 3D reconstruction. This can cause problems when the 3D reconstruction is used for proximity detection and collision avoidance by a space vehicle working around another space vehicle. As such, we would like the automatic ability to recognize when a hole in a 3D reconstruction is in fact not a hole, but is a place where reconstruction has failed. Once we know about such a location, methods can be used to try to either more vigorously fill in that region or to instruct a space vehicle to proceed with more caution around that area. Detecting such areas in earth orbiting objects is non-trivial since we need to parse out complex vehicle features from complex earth features, particularly when the observing vehicle is overhead the target vehicle. To do this, we have created a Space Object Classifier and Segmenter (SOCS) hole finder. The general principle we use is to classify image features into three categories (earth, man-made, space). Classified regions are then clustered into probabilistic regions which can then be segmented out. Our categorization method uses an augmentation of a state of the art bag of visual words method for object categorization. This method works by first extracting Pyramid-Histogram-Of-visual-Words (dense SIFT like) features, which are computed over an image and then quantized via KD Tree. The quantization results are then binned into histograms and results classified by the PEGASOS support vector machine solver. This gives a probability that a patch in the image corresponds to one of three categories: Earth, Man-Made or Space. Here man-made refers to artificial objects in space. To categorize a whole image, a common sliding window protocol is used. Here we utilized 90 high resolution images from space shuttle servicing missions of the international space station. We extracted 9000 128x128 patches from the images, then we hand sorted them into one of three categories. We then trained our categorizer on a subset of 6000 patches. Testing on 3000 testing patches yielded 96.8% accuracy. This is basically good enough because detection returns a probabilistic score (e.g. p of man-made). Detections can then be spatially pooled to smooth out statistical blips. Spatial pooling can be done by creating a three-channel (dimension) image where each channel is the probability of each of the three classes at that location in the image. The probability image can then be segmented or co-segmented with the visible image using a classical segmentation method such as Mean Shift. This yields contiguous regions of classified image. Holes can be detected when SFM does not fill in a region segmented as man-made. Results are shown of the SOCS implementation finding and segmenting man-made objects in pictures containing space vehicles very different from the training set such as Skylab, and the Hubble space telescope.

Keywords: Classification, Segmentation, 3D Reconstruction, Structure from Motion, Hole Finding, Hole Filling

1. INTRODUCTION

1.1 Problem Overview

There is growing interest in creating autonomous spacecraft that can rendezvous with other spacecraft and perform repair and reconstruction on the target spacecraft. An issue that arises is that as a spacecraft approaches and works on the target object, the craft or the operator must be aware of the proximity and extent of the target craft to avoid collisions between sensitive instrumentation. Additionally, actions in space frequently need to be carefully choreographed. If we can inspect the target craft before executing a task, we can optimize the actions. One approach for allowing us to

1 Corresponding Author: tnmundhenk@hrl.com

visualize the target craft and to know its extent and proximity is to use Shape from Motion (SFM) [3] to create a 3D model from images of the target craft taken by the servicing vehicle. This method can then be used to help us avoid collisions and to inspect the other craft visually without necessarily needing to move all around it.

Shape from Motion approaches create a 3D model by solving for camera pose by matching key features between camera views. These methods can vary by which features are matched, how the pose and perspective problem is solved and how points are filled in between matched key points. A common method is to extract Scale-Invariant Feature Transform (SIFT) features [4] then use RANSAC [5] to find inlier matches between images. A gradient method such as Levenberg-Marquardt [6] is used to minimize the error in pose and find the transformation matrix which maps key point in one image to another. Series of images can then be bundled together with known camera poses and 3D-mapped key points. This bundling creates a sparse point cloud, which is useful for matching camera viewpoints, but is not visually appealing. A second method such as Clustering Views for Multi-view Stereo (CMVS) [7] can be used to fill in stereo points between the key points as well as perform matching between 3D regions which SFM was unable to connect. This last step is a topic of various ongoing research efforts with many competing methods such as probabilistic voxel methods [8, 9] and surface mapping [1, 10, 11].

An issue that comes up during the final step of reconstruction is that there are frequently holes in the 3D model. If the surface is visible, this is typically the result of low feature or specular surfaces. However, any optical anomaly which prevents stereo feature matching can be the cause. An example of such holes can be seen in Figure 1. When these holes are present in the data, the question then becomes when does the hole belong there and when is the hole due to a failure in the 3D constructing software? The answer to this question determines a variety of courses of action. If we believe the hole belongs there, then we can leave the 3D construction as it is. If the hole does not belong there, then we can try other 3D reconstruction methods locally in an attempt to fill the hole. For instance, we can project a plane over the hole and overlay the image. We might also apply a different reconstruction method locally that might otherwise be more expensive such as using a probabilistic voxel method [2]. Alternatively, even if we cannot reconstruct the region of a hole of data, knowledge of its presence can be used to instruct a servicing vehicle to avoid said region. Thus, it is helpful to know when a hole is a hole and when it is not.

What we desire is a method to determine where holes in a 3D model of the target vehicle may reside. Additionally, we may desire to know what kind of hole we have. Is the hole caused by specularity or by a flat low feature region? By knowing such things, we may further hone the method we use to attempt to fill the hole in our 3D data. Additionally, the method we choose should be efficient enough that frame rate solutions should be feasible. We would like to be able to create our 3D model fast enough to prevent collisions between spacecraft, as such we would like to be able to find holes efficiently as well. In the remainder of this article, we present our new method for finding holes, which relies on identifying regions containing spacecraft parts within an image, and show that this method can correctly identify previously unseen spacecraft.

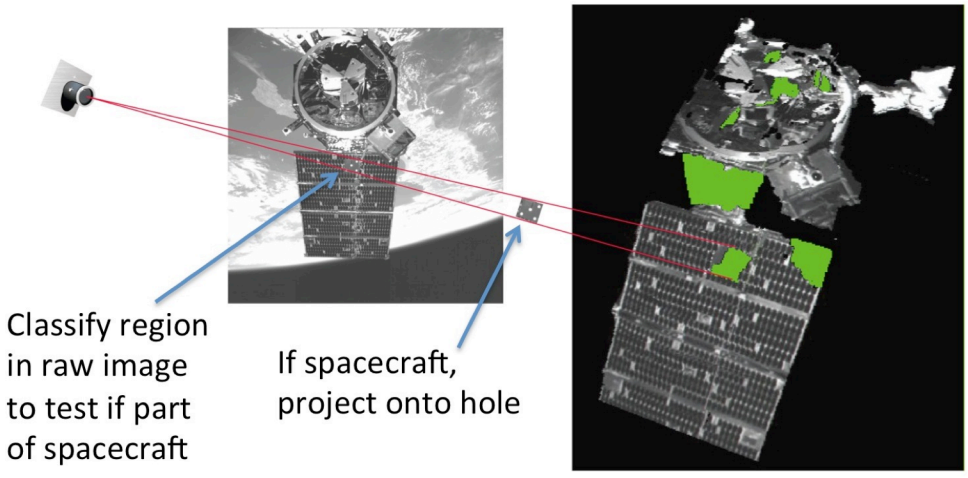

Figure 1. Classifying regions within a 2D image for being part of the target spacecraft enables the filling of a hole by projecting a region within the raw image onto the 3D model. Areas highlighted in green are places where holes incorrectly appear in the 3D model. The 3D reconstruction (Right) was kindly provided by Dr Jan Frahm and generated with a modified algorithm in the spirit of [17].

1.2 Space Object Classification and Segmentation (SOCS)

Our method for finding holes is to classify regions in an image and then to segment those regions. So, if we have a hole in our 3D data set that corresponds to an image region that has been classified as *man-made*, then we know that we probably have a failure of the 3D reconstruction method. If we have an area with 3D data missing which is classified as *earth*, then we can surmise that it's a hole which should be there if the camera is pointed towards earth. However, if the camera is oriented towards space, earth features most likely suggest a specular region that is reflecting the surface of the earth. The same heuristic can be applied to regions identified as *space*. If the camera is pointed towards the earth, a region identified as space is most likely a reflection. Thus, in general we want to be able to identify three categories of objects in images. These are Earth, man-made and space. This set is sufficient for detecting specular surfaces or proper holes in the target craft given different vantage points about the target spacecraft.

Briefly, our method works by learning the features associated with the three categories we are interested in. Blocks within in image are given a probability of belonging to any of the categories. A segmentation method is then used to find the continuous regions of a feature over an object. This segmentation allows such regions to have probabilities assigned to them. As a result, we have a set of image segments with probabilities of membership in each of the categories. We assert that several available methods are suitable for this segmentation. However, the classification method must be able to deal with multiple viewpoints and arrangements of features. It must also return a score which can be interpreted as a probability. This will be covered in more detail later on.

2. METHODS

In this section, we present our methods for space object classification, training of the classifier, running the classifier, and image segmentation.

2.1 Space Object Classification

The SOCS method is trained on sets of real space images taken around the International Space Station (See examples in Figure **2**), from which image patches were extracted (Section 2.2). We split the image patches into three categories: Earth, Man-made and Space. To deal with the large amount of viewpoint and illumination variation, we need a categorization method that is robust to these properties. As such, a method that is based on feature statistics, but is insensitive to the arrangement of these statistics is advantageous; for example, a solar panel contains certain kinds of lines and textures, but these lines and textures can be arranged in many different ways. Additionally, if we are looking at the solar array from one angle, lines that run along the array may appear diagonal in the image, while in another image, they are horizontal. So, we are less concerned with where features may occur, but are more concerned with the statistical composition of these features. A good method for classification of objects in this way is the VLFeat object categorizer [12], which is a freely-available open source classification software. It works by extracting Dense Scale-Invariant Feature Transform (DSIFT) features in windowed regions of an image. We used a variant of DSIFT, namely the so-called Pyramid Histogram Of visual Words (PHOW) [14]. A sparse dictionary is formed by K-means clustering. This dictionary is used to classify the dense SIFT features. Once classified, we tally up the classified features per bin. This yields a histogram of classified features. Using a Chi-Squared kernel, we train a support vector machine (SVM) to give the probability of a windowed region being a sample of one of the three categories. The probability is derived from the dot product and reflects the distance of histogram to one side or the other from the separating hyper plane. The SVM probability is obtained by applying a hyperbolic tangent function to the raw SVM score which forces it to range from -1 to 1.

2.2 Training the SOCS classifier

The SOCS classifier works by using a sliding window protocol to determine if a region matches the statistics of training regions. So training involves creating this region classifier. For training, we selected images of the International Space Station (ISS) because it has a high diversity of man-made objects in high resolution. From a set of 90 images of the ISS, we randomly extracted several thousand 128x128 training patched. We then inspected these and sorted them by hand into one of the three categories. This yielded 9000 patches in all. Using a subset of 6000 training patches (2000 per class), patches are converted into L*a*b color space. Dense SIFT features are extracted at every other pixel location in the training patch over each of the three color channels. The SIFT descriptor returned had a standard 128 features per

color channel. The color value in L*a*b space was added into the descriptor to add in first-order statistics. Since color is very revealing in this type of classifier, inclusion seems prudent. The color range is adjusted so that it has the same range as the SIFT features. In all, this returns a feature vector with 387 values. The feature dictionary is created from a random subset of 102,400 features over the set of training patches. Using k-means clustering, we obtain a dictionary with 1024 feature words. We then form a kd-tree classifier from the k-means result to reduce the dictionary lookup time to log order. Once we have our dictionary of features, we train the support vector machine classifier. This training is done by adding up the number of feature words in the training patch and normalizing by the total number of words. The histogram created is pre-processed by a Chi-squared kernel and fed into a PEGASOS SVM classifier [13]. Since this classifier works with two classes, we train three classifiers, one for each class. This yields a fairly simple dot product classifier for each of the three classes that allows us to solve the three-class problem. For more details on this method, readers are directed to [12, 14].

2.3 Running the SOCS classifier

Figure 3 shows a block diagram of how the classifier runs. The SOCS classifier uses a standard sliding window protocol. The window is the same size as the training patches (128x128). Note that because the histogram is normalized, we can in theory use other window sizes. The sliding window is moved in even steps. Step sizes between 16 to 64 pixels seems effective. Smaller steps yield a more precise result, but take longer to compute. Each 128x128 window is classified by the SVM as being one of the three trained classes (Earth, man-made, space). The return value is a score where the higher the score is, the stronger the confidence is in identification. A negative number can be returned for each classifier as well. This number signifies the classifier's belief that a patch is *not* the object. The score is squashed by a tangential sigmoid (hyperbolic tangent) which forces scores to range from -1 to 1. To force the range between 0 to 1, we add 1 to the score and then divide by 2. For each patch, this means three scores are returned, one for each class. The patch score is stored in an image at the location of the patch in the original image. For example, given an image which is 1525x1904, if we use a 16 pixel step with a 128x128 sized patch, the sliding window protocol will return a score image that is

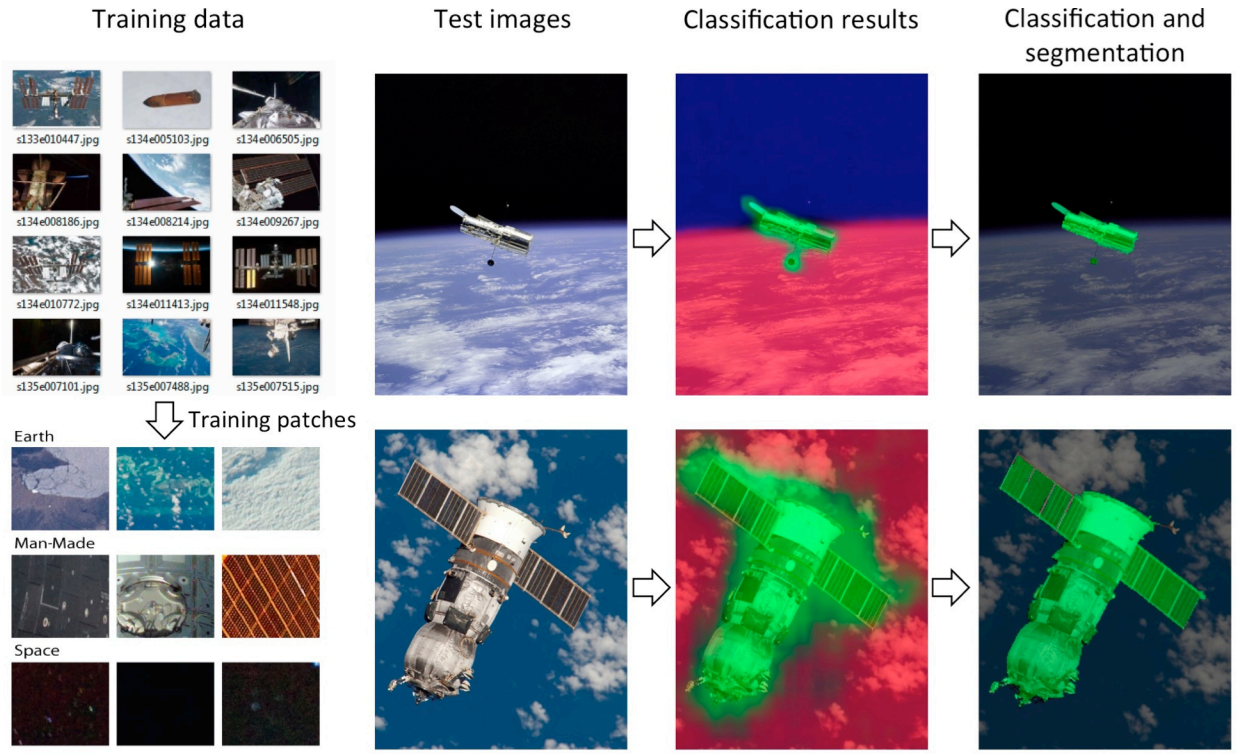

Figure 2. The top left corner shows examples from the ISS images used to extract training patches. Below it are examples of the training patches that we used to train the classifier. Using the classifier, we derive a probability map over the image that a location can be classified as Earth, man-made or space. Then we use a segmentation step to trim and combine the probability map into a region.

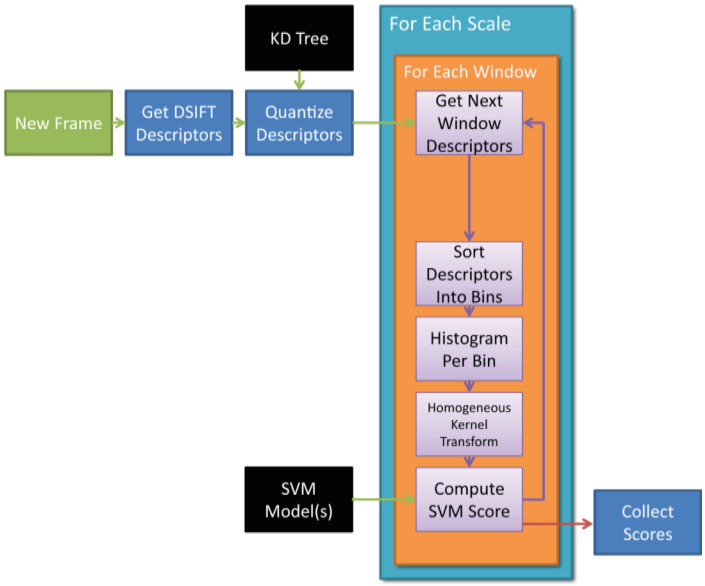

Figure 3. Block diagram showing the sliding window protocol for classifying patches in an image.

88x111 with three channels for the score of each class of object. If we scale this image back to the size of the original image, it gives us a raw score over the entire image of regions belonging to one class or the other (Figure 2, column "Classification results"). This score image roughly corresponds to objects in an image. The next step is to translate this rough score image into segments which match more tightly to the actual extent of objects in the image.

2.4 Running the SOCS segmenter

For this experiment, we used a Mean-Shift segmentation method as our base segmenter [15, 16]. This algorithm was chosen for convenience (other segmenters may work as well or better). For computational efficiency, we reduced the size of the raw input images (between 15% to 50% of their original size). The segmenter returns a list of labeled regions. For each labeled segment, we will find the mean probability for each of the three classes. The maximum probability determines the final classification of that labeled segment. So given n pixels in a segment with the same label l (a contiguous region in the image) with scores from the sliding window classifier s for each class θ the unnormalized probability for a segment belonging to that class can be loosely expressed as:

$$P(\theta|l) = \frac{\sum_{i=0}^{n} s_{\theta}^{i}}{n} \quad (1)$$

The class that the segment belongs to is then the max of the three $P(\theta|l)$ scores. This has the effect of smoothing out scores along continuous feature segments and creating reasonable border regions. Figures 2 and 5 show the segmented probability maps of the class "man-made", which are the output from this step. We can apply a second criteria to the classification: add an unknown category if no class scores above a certain threshold. In our embodiment, we set this probability threshold to 0.5.

An overview of the process flow for image classification and segmentation is shown in Figure 4. The numbers in this figure refer to the following processes:

(1) We input the prior trained models for each class with the L*a*b color space model. These will run with a sliding window protocol to identify matches at each location to each of the object classes (See Figure 3).

(2) The output from the sliding window protocol is an image of scores. In this example, there are three images, one for each category. If the score > 0, then it is considered a match at that location. The score image corresponds directly to the original image in scale.

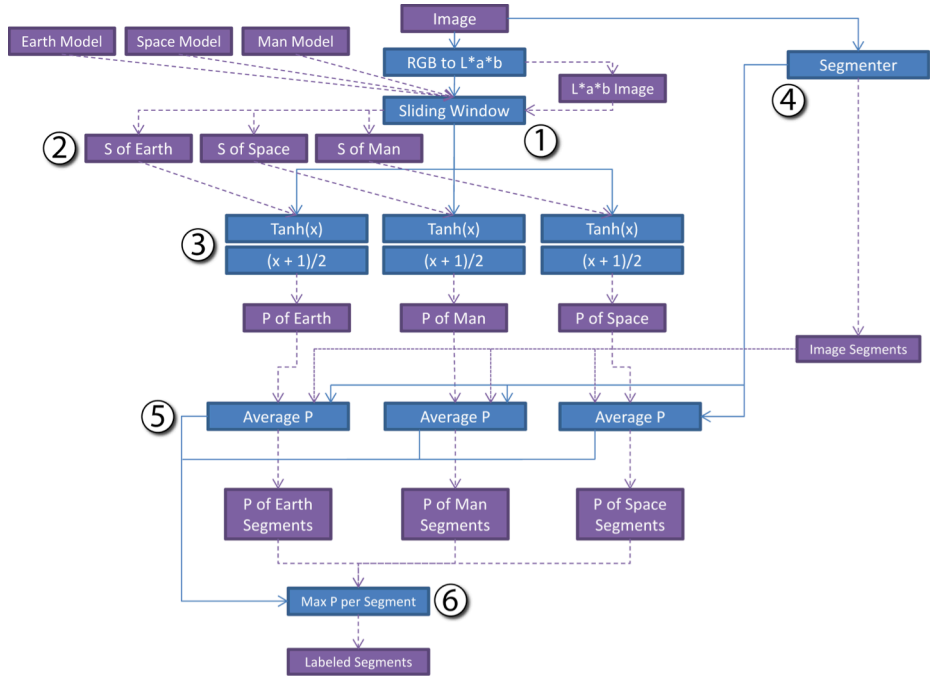

Figure **4. O**verview block diagram of the image classification and segmentation. Instruction paths are solid blue; data paths are dashed purple. Here, the diagram shows as example three image classes: Earth, Man-Made, Space. See text for an explanation of the numbered processes.

(3) We condition the score image into a probability image by squashing the scores with a hyperbolic tangent, which forces the scores to range from -1 to 1. Then, we add 1 and divide by 2. This forces the scores to range from 0 to 1, which we interpret as a probability. The output is a probability image (See Figure 5 "Classification" or Figure 2 "Classification Results").
(4) We input the image into the segmenter. The output is a map of segments based on simple features like color (see Figure 5 "Segmentation").
(5) For each segment, we find the average probability in its region for each category by combining the probability images and the segmented image. The output is a segmented probability map.
(6) For each of the segments, the max probability is the label given to that segment. However, if all probabilities for a segment are less than 0.5, then it is labeled as a category of unknown, since none of the models matched with a positive score. The output is the final labeled image of segments (See Figure 5 "Classified Segments").

3. RESULTS

Our SOCS method could successfully identify a wide variety of space objects (Figures 2, 5, and 6). Here, we used a variety of spacecraft that are very different from the training data. The combination of segmentation and classification resulted in tight boundaries around the spacecraft. Thus, SOCS would be able to identify regions in which holes should be filled in the 3D reconstruction. On the 3000 patches of the test set, we observed a 96.8% accuracy for classification. Errors are mainly constrained to border regions where earth meets space and over exposed pictures of space which have an amber appearance. However, since we compute the average classification score over a segmented image region, the effective classification performance is much higher.

4. DISCUSSION

Our objective is to find holes in 3D reconstructions of unknown spacecraft from images. To achieve this objective, we developed a novel method to find the entire tight region of a spacecraft within an image. Our method combined a classifier of space image patches with a segmentation method. As a result, we could accurately separate a wide variety of

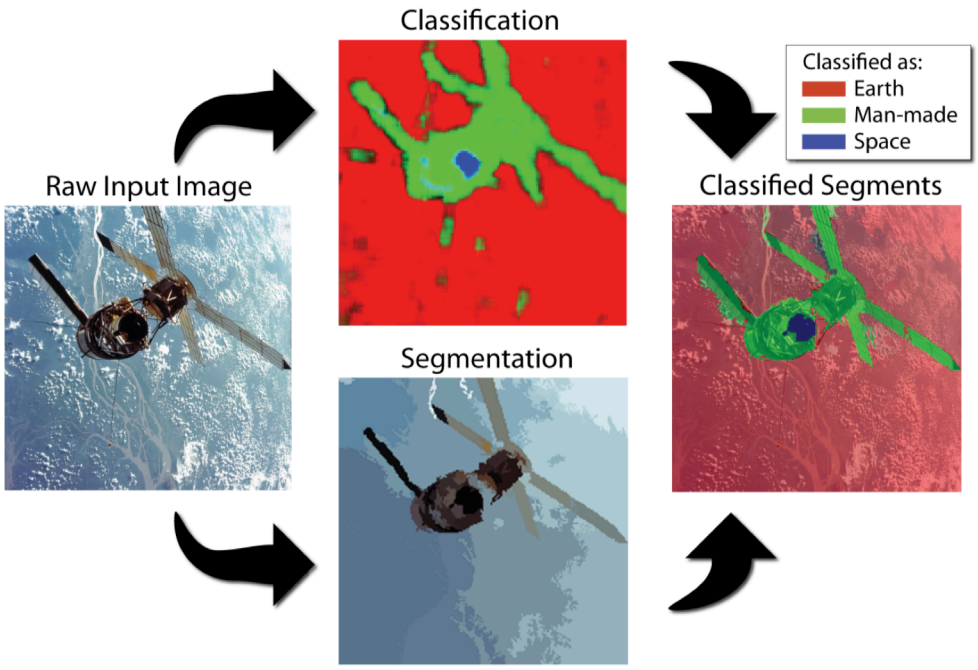

Figure 5. Classification is computed as a parallel step to computing segmentation. They are then brought back together by taking the mean score for the probability map within each segment, resulting in tighter boundaries of the classification. In this example, part of Skylab appears as space. It is unclear if the area is specular or just very dark. However, in the hole filing paradigm that we have proposed, a camera pointed towards Earth could relabel anything classified as space into specular man-made.

spacecraft from Earth background, even though the classifier was trained only on ISS images. This segmentation could be compared with the 3D reconstruction to identify holes. Our results are mainly qualitative, but they do show the feasibility of the method we have proposed. A more thorough evaluation is left for future work. In the following, we discuss the generalization of our method and specular region classification and end with the highlights of our results.

4.1 General statistics seem revealing in special cases

Our solution was feasible partly thanks to the constrained statistics of objects in our training set. We could obtain a rather thorough set of images for space and Earth, which are rather inclusive. For very broad categories with fairly crisp statistical boundaries, the method may work. So the method may also work in terrestrial settings for looking for man-made v. natural. A problem that we observed with the method is that it sometimes picked out man-made structures visible from space such as road networks as man-made and not Earth. This is an ambiguous error in some respects. It runs contrary to the task at hand, but the method may have just learned identification too well. A full analysis would be interesting to see if it is generalizing that well. If this is the case, then it may suggest that one can use it for aerial imagery to spot man-made objects.

4.2 Specular region classification

In the Skylab and Apollo command module, there are man-made objects classified as space which are not space. The Apollo module is highly reflective and it is difficult to see how we would get the correct answer for that region. As humans we can make a gestalt completion of the object to know where there is a reflection. However, if we know the angle of the space craft, we can classify these two examples as a specular related error since they lay against the backdrop of the earth. However, the Apollo landers reflection lays at the horizon of earth and space. If a space reflection overlaps with a space backdrop, then we would have an error we would not be able to detect. Thus, some specular regions may be very difficult to detect with any method.

4.3 Highlights of our results

Our new SOCS method demonstrated the following properties:

(1) Boundaries of spacecraft can be found against complex distractors sufficiently to determine real holes v. illusory holes.

(2) The method can generalize to spacecraft it has never seen before.

(3) The method can generalize to earth like features different from those in the training set.

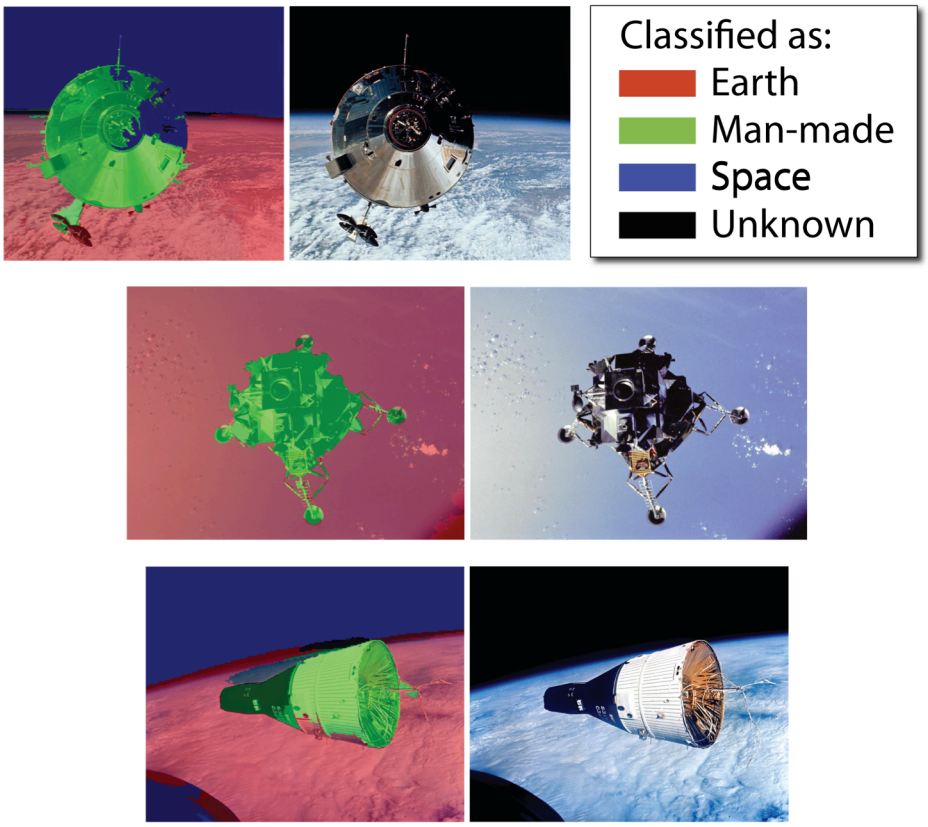

Figure 6. Generalization is demonstrated on three man-made objects that are from a different era than the ISS. The pictured spacecraft are: (Top) Apollo Command Module (Middle) Apollo Lunar Lander (Bottom) Gemini Space Craft. Part of the Apollo module is labeled as space. This mislabeling is reasonable since it is a highly specular surface reflecting space.

REFERENCES

[1] J.-M. Frahm, P. Georgel, D. Gallup, T. Johnson, R. Raguram, C. Wu, Y.-H. Jen, E. Dunn, B. Clipp, S. Lazebnik, and M. Pollefeys, "Building Rome on a Cloudless Day," presented at the ECCV, 2010.

[2] A. Miller, V. Jain, and J. L. Mundy, "Real-time rendering and dynamic updating of 3-d volumetric data," in *The Fourth Workshop on General Purpose Processing on Graphics Processing Units* 2011.

[3] N. Snavely, S. M. Seitz, and R. Szeliski, " Photo Tourism: Exploring image collections in 3D. ," in *SIGGRAPH*, 2006.

[4] D. G. Lowe, "Object recognition from local scale-invariant features," in *Seventh IEEE International Conference on Computer Vision*, 1999, pp. 1150-1157.

[5] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM,* vol. 24, pp. 381-395, 1981.

[6] K. Levenberg, "A Method for the Solution of Certain Non-Linear Problems in Least Squares," *Quarterly of Applied Mathematics* vol. 2, pp. 164–168, 1944.

[7] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski, "Towards Internet-scale Multi-view Stereo," presented at the CVPR, 2010.

[8] A. Miller, V. Jain, and J. L. Mundy, "A heterogeneous Framework for Large-Scale Dense 3-d Reconstruction from Aerial Imagery," *IEEE Transactions on Parallel and Distributed Systems,* vol. IN PRESS, 2013.

[9] M. I. Restrepo, B. A. Mayer, A. O. Ulusoy, and J. L. Mundy, "Characterization of 3-D Volumetric Probabilistic Scenes for Object Recognition," *IEEE Journal of Selected Topics in Signal Processing,* vol. 6, pp. 522-537, 2012.

[10] B. Clipp, R. Raguram, J.-M. Frahm, G. Welch, and M. Pollefeys, "A Mobile 3D City Reconstruction System," presented at the IEEE VR workshop on Cityscapes, 2008.

[11] C. Wu, B. Clipp, X. Li, J.-M. Frahm, and M. Pollefeys, "3d model matching with viewpointinvariant patches (vip)," in *CVPR*, 2008.

[12] A. Vedaldi and B. Fulkerson, "VLFeat: An Open and Portable Library of Computer Vision Algorithms," ed, 2008.

[13] Y. Singer and N. Srebro, "Pegasos: Primal estimated sub-gradient solver for SVM," presented at the ICML, 2007.

[14] A. Bosch, A. Zisserman, and X. Munoz, "Image classifcation using random forests and ferns," in *ICCV*, 2007.

[15] *EDISON*. Available: http://coewww.rutgers.edu/riul/research/code/EDISON/

[16] D. Comaniciu and P. Meer, "Mean Shift: A Robust Approach Towards Feature Space Analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 24, pp. 1-18, 2002.

[17] David Gallup, Jan-Michael Frahm, Marc Pollefeys, "Piecewise Planar and Non-Planar Stereo for Urban Scene Reconstruction", IEEE Conference Computer Vision and Pattern Recognition (CVPR) 2010