Rapid Three-Dimensional Orientation Resolver (RAPTOR)

Sam Kelly, Department of Computer Science, Dickinson College, Carlisle PA 17013
NREIP, Naval Research Laboratory, Adaptive Systems Section

1. Introduction

In this manuscript we present RAPTOR (Rapid Three-Dimensional Orientation Resolver), which is a novel pipeline for inferring the 3D distance and orientation of known classes of rigid objects from 2D image data using machine learning and/or deep learning[1]. For each object class our system supports, we train a separate neural network that becomes an expert on that class. Each network takes 2D pixel data representing an instance of that network's supported object class as input, and learns to infer from this pixel data the 3D orientation and distance of the object appearing in said input image with respect to the camera. Individual networks are trained on thousands of 2D renderings of a man-made 3D model for the corresponding object rendered at a variety of random camera angles, distances, and lighting conditions. Our system is intended as a proof-of-concept, and as a first step towards a larger, future project that would combine RAPTOR with sate-of-the-art object recognition (i.e. OverFeat [1]) and 2D-to-3D mesh generation and 3D reconstruction techniques (i.e. Cornell's Make3D [2]) to infer in real time a *full* 3D understanding of a tactical environment merely from 2D video or image data.

1.1 Motivations

Artificial agents in modern 3D video games are tactically formidable because they are able to leverage a wealth of game state and 3D scene information that would be impossible or incredibly difficult to infer visually in an uninformed fashion. General techniques exist that attempt to infer full 3D depth information from 2D video or image data, however these techniques tend to be over-sensitive to lighting conditions and optical illusions, often leading to dramatic inaccuracies and deformations in the resulting 3D scene. While depth sensors and/or LIDAR can help alleviate this problem, there are many situations where accurate depth information is required, but only 2D imagery is available. Existing

1 Note: If RAPTOR ends up being too slow, we will have to come up with a different acronym.

techniques tend to focus on generating 3D mesh for surrounding terrain and buildings rather than figuring out the 3D position and orientation of common world objects such as vehicles, signs etc., for which accurate man-made 3D models are readily available. Next-generation tactical systems should be able to (1) infer from 2D imagery an accurate 3D understanding of the surrounding terrain and obstacles; (2) recognize and *label* nearby rigid tactical objects, such as ground vehicles, ships, planes, weapons, ordinance, helmets, etc; and (3) correctly *orient* and *embed* full, unoccluded 3D meshes of these objects in the resulting 3D scene. In principle, a system capable of meeting all three of these requirements in real time would have roughly the same wealth of scene information available to artificial agents in modern video games, potentially allowing such a system to outperform similar existing systems that operate in real-world domains.

1.2 Related Work

Most prior work in this space either focuses on 3D template matching merely as an object detection or object recognition problem, tries instead to solve the much harder problem of inferring per-pixel depth information from a 2D image in a class-insensitive fashion, and/or doesn't make use of machine or deep learning. Additionally, many papers exist that focus on 3D reconstruction, or rather, constructing 3D meshes based on thousands of 2D images of an object or environment, but we were not able to find any existing work that tries to take advantage of man-made 3D models to diminish or mitigate this process when reconstructing a full 3D scene. Thus in general, there is a wealth of existing work that attempts to solve (1) and (2), but very little work has been conducted on (3), which boils down to inferring the 3D depth and orientation of a known (already classified) object based on 2D pixel data. To our knowledge, no one has attempted to apply neural networks or deep learning to (3). A major reason for this might be the lack of relevant training data – while it might be feasible to take thousands of images of a ball or a teapot from varying distances and angles, doing so would be much more difficult in the case of large real-world object classes such as cars, ships, and planes. Typically there is just enough available data for these sorts of objects for a network to learn recognition, but not enough to learn to infer 3D distance and orientation from a 2D image.

OverFeat [1] combines an efficient scale-invariant feature extraction pipeline with convolutional neural networks to facilitate the recognition of thousands of classes of objects in 2D images and video. The authors of [3] use a

similar technique to perform object classification, also making use of convolutional neural networks in their pipeline, since CNNs are particularly suitable for image data. The authors of [4] use a unique melding of generative and discriminative layered deep belief nets to recognize 3D objects based on the NORB database of stereo-pair images (i.e. depth information is already part of the input). LINE-MOD [5] provides a gradient-based approach for real-time 3D object instance recognition that does not rely on neural networks or machine learning. All of these approaches concern themselves merely with object recognition, and do not attempt to address object distance or 3D orientation as RAPTOR does.

Cornell's Make3D [2] provides impressive results on the incredibly difficult problem of generating a 3D mesh of a scene from a single monocular image. In [6], a similar technique is used whereby semantic cues are detected in the 2D image as a preprocessing step to help take the burden of differentiating buildings from ground, and ground from sky, etc., off of the learning algorithm. The authors of [7] use a coded lens and exploit image focal length properties to obtain accurate per-pixel distance estimations from a conventional camera. In general the work in this category focuses on the much harder problem of generating 3D meshes or per-pixel depth information from a 2D image in a bottom-up fashion rather than limiting this to a problem of orienting an existing 3D model for an already-identified object class in a 3D scene based on a 2D image, as is the case with RAPTOR.

2.1 RAPTOR Overview

RAPTOR is designed to take a region of a 2D image that corresponds with a known object class as input. For output, RAPTOR provides the 3D distance and 3D orientation of said object with respect to the camera. To accomplish this, we intend to test a number of machine learning and deep learning architectures, including but not limited to standard feed-forward artificial neural networks, deep belief networks, convolutional neural networks, and other more exotic architectures possibly including sun-product networks, so we can determine which architecture is best suited for learning class-sensitive 3D distance and orientation.

RAPTOR will take image data as input and produces four continuous values as output, so technically this is a function approximation problem rather than a problem that can be solved directly using generative or discriminative methods.

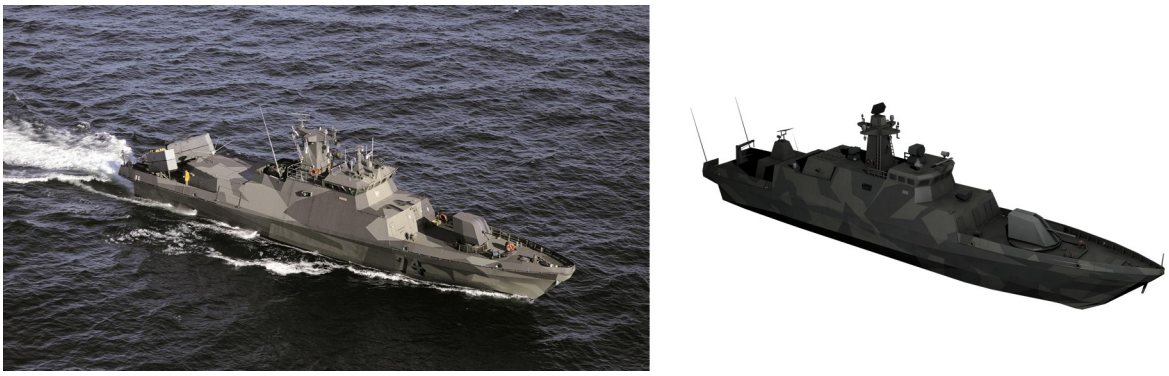

Figure 2.1: A real-world picture of a Hamina-class missile boat shown next to a 2D rendering of a properly oriented man-made 3D model of the same missile boat. Since the 3D model is properly oriented with respect to the picture, we can easily calculate important, non-trivial state information about the boat shown in the picture, such as distance from camera, movement speed, trajectory, proximity to other objects, etc. The goal of RAPTOR is to be able to perform orientations such as these on the fly by training on 2D renderings of 3D models such as the one shown to the right.

While there is ample existing work that applies standard feed-forward artificial neural networks to function approximation problems, there is very little existing work that attempts to apply deep learning to such problems, and as far as we can tell, none of these attempts to learn both 3D distance and orientation from renderings of 3D models in a class-sensitive fashion. While it is relatively uncommon compared to classification, function approximation is alive and well in the deep learning community. For example, [8] uses deep learning techniques to tackle the problem of playing seven unrelated ATARI games as a function approximation problem based only on image input taken from game screen-shots. We intend to use techniques similar to those used in this paper to coax deep learning architectures to learn the distance and orientation estimation problem faced by RAPTOR.

2.2 Architecture

Given a known object class C, and pixel data P known to correspond with an instance of C, a RAPTOR instance trained on C is defined as some sort of artificial neural network that is able to take P as input and return as output a scalar D representing the 3D distance from the camera to the object, and a vector R which represents how the object is oriented (rotated) in 3D space with respect to the camera. Using D and R, one can correctly orient and embed a man-made 3D model of C in a 3D scene. Each RAPTOR instance only handles objects that are of

the class that it was trained on. Thus each instance specializes as an expert on resolving the 3D distance and orientation of instances only of its designated object class.

As mentioned before, we do not yet know the exact network architecture we will use for each RAPTOR instance, but there are a number of possibilities we intend to evaluate. Since no existing work seems to have tackled this problem directly, we do not yet know whether the problem is difficult enough to require deep learning, or if 3D distance and orientation of a specific object class can actually be solved by a standard feed-forward artificial neural network or an SVM. Likewise, we intend to experiment with a number of different input/output representation schemes and feature extraction pipelines to fine-tune our results.

We are also limited in the sense that we cannot use scale-invariant feature extraction procedures, as we need scale to preserve distance information. Because of this, we have to support both situations where the object is very near and its pixels take up the entire viewport, and situations where the object is very far away, so that its pixels take up a very small portion of the viewport. This implies that our input grid needs to be the size of the viewport, which would be a major problem because we would lose a large amount of resolution because we would likely have to down-scale the entire input grid in a uniform fashion. One method we have come up with for alleviating this problem is to always scale the object pixels to fit the input grid (thus destroying distance information), but then to encode the scaling factor we used and include this along with the input. In this way we retain maximal pixel resolution, but RAPTOR will still be able to calculate distance accurately, at least in theory.

2.3 Training

Most systems in 3D reconstruction and object recognition rely on real-world images to facilitate most of their training, however in the case of RAPTOR it would be too difficult to acquire the number of images that would be needed, especially considering the large size of some of the objects we want to work with (i.e. naval ships). To combat this, we came up with the idea of training on renderings of 3D models of our objects at arbitrary distances and orientations. This means that for RAPTOR to be able to learn any given object, there must be a man-made 3D model available for that object. It remains to be shown, however, whether learning on renderings will generalize such that RAPTOR can make correct estimations based on real-world images. There are also obstacles that we will eventually have to deal with involving different camera focal lengths, etc. That

said, there do exist expensive 3D rendering pipelines such as Eon Vue that are known for their ability to create seamlessly photo-realistic renderings including realistic oceans, waves, weather, foliage, etc. Likewise, filters could be used during rendering to simulate the focal properties of a particular camera, and this iis actually a common practice in certain CAD environments. We do not have plans to work with any of these pipelines for this project, however, since this project is intended more as a proof-of-concept. If we are able to obtain decent results on purely rendered data, then we will also asses how well RAPTOR adapts to real-world image data, though doing so will require a good degree of work on our part – we will need to manually orient 3D models in 2D images and add the proper annotations, and this can take up to an hour per image.

The training process for an object class *C* can be broken down as follows: Once a suitable (i.e. textured) 3D model for *C* has been acquired, we will use a Python script to hook into Blender 3D's rendering pipeline (Blender is written entirely in Python and has an extensible design allowing for task automation using Python). This will allow us to create high quality ray-traced renderings with realistic lighting and textures without having to write our own pipeline for doing so. Our python script will allow us to load a 3D model and randomize camera position, angle, and lighting (and possibly texture) properties. Thus we can create a virtually infinite set of training data for any given 3D model. Distance and rotation are controlled by the script, so this information will be known a priori meaning testing will be incredibly easy. As mentioned before, we also plan to supplement this data with a sparse set of real-world image data to improve RAPTOR's ability to generalize. Rendered training images will be annotated with their respective distance and orientation information.

3.1 Conclusion

The aim of RAPTOR is to provide an accurate method for embedding already-created 3D models in a scene based on 2D pixel data representing a known object class in such a way that distance to camera and 3D orientation are resolved. While similar problems have been studied extensively in the existing literature, to our knowledge no one has tried to tackle 3D scene reconstruction as a problem of embedding existing 3D models, and few have focused on 3D reconstruction in a class-sensitive fashion to begin with. Likewise, there are very few cases in the existing literature where researchers exploit renderings of 3D models to supplement a sparse set of 2D image data. If RAPTOR works well, we will likely try to publish our results since this technique has the potential to

positively impact the computer vision community and the deep learning community, not to mention the potential military, law enforcement (i.e. traffic violation detection), and surveillance applications RAPTOR could also be used for (i.e. a surveillance system could monitor a particular object's position in 3D, or trigger an alert if one class of object comes too close to another class of object, etc.).

Works Cited

[1] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, “OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks,” *arXiv Prepr. arXiv1312.6229*, 2013.

[2] A. Saxena, M. Sun, and A. Y. Ng, “Make3d: Learning 3d scene structure from a single still image,” *Pattern Anal. Mach. Intell. IEEE Trans.*, vol. 31, no. 5, pp. 824–840, 2009.

[3] R. Socher, B. Huval, B. P. Bath, C. D. Manning, and A. Y. Ng, “Convolutional-Recursive Deep Learning for 3D Object Classification.,” in *NIPS*, 2012, pp. 665–673.

[4] V. Nair and G. E. Hinton, “3D Object Recognition with Deep Belief Nets.,” in *NIPS*, 2009, pp. 1339–1347.

[5] S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, and V. Lepetit, “Gradient response maps for real-time detection of textureless objects,” *Pattern Anal. Mach. Intell. IEEE Trans.*, vol. 34, no. 5, pp. 876–888, 2012.

[6] B. Liu, S. Gould, and D. Koller, “Single image depth estimation from predicted semantic labels,” in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, 2010, pp. 1253–1260.

[7] A. Levin, R. Fergus, F. Durand, and W. T. Freeman, “Image and depth from a conventional camera with a coded aperture,” *ACM Trans. Graph.*, vol. 26, no. 3, p. 70, 2007.

[8] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, “Playing Atari with Deep Reinforcement Learning,” *arXiv Prepr. arXiv1312.5602*, 2013.