# ABSTRACT


Title of dissertation: From Form to Function:
Detecting the Affordance of Tool Parts
using Geometric Features
and Material Cues

Austin O. Myers, Doctor of Philosophy, 2016

Dissertation directed by: Professor Yiannis Aloimonos
Department of Computer Science



With recent advances in robotics, general purpose robots like Baxter are quickly becoming a reality. As robots begin to collaborate with humans in everyday workspaces, they will need to understand the functions of objects and their parts. To cut an apple or hammer a nail, robots need to not just know a tool's name, but they must find its parts and identify their potential functions, or affordances. As Gibson remarked, "If you know what can be done with a[n] object, what it can be used for, you can call it whatever you please."

We hypothesize that the geometry of a part is closely related to its affordance, since its geometric properties govern the possible physical interactions with the environment. In the first part of this thesis, we investigate how the affordances of tool parts can be predicted using geometric features from RGB-D sensors like Kinect. We develop several approaches to learn affordance from geometric features: using superpixel based hierarchical sparse coding, structured random forests, and convolutional neural networks. To evaluate the proposed methods, we construct a large

RGB-D dataset where parts are labeled with multiple affordances. Experiments over sequences containing clutter, occlusions, and viewpoint changes show that the approaches provide precise predictions that can be used in robotics applications.

In addition to geometry, the material properties of a part also determine its potential functions. In the second part of this thesis, we investigate how material cues can be integrated into a deep learning framework for affordance prediction. We propose a modular approach for combining high-level material information, or other mid-level cues, in order to improve affordance predictions. We present experiments which demonstrate the efficacy of our approach on an expanded RGB-D dataset, which includes data from non-tool objects and multiple depth sensors. The work presented in this thesis lays a foundation for the development of robots which can predict the potential functions of tool parts, and provides a basis for higher level reasoning about affordance.

# From Form to Function: Detecting the Affordance of Tool Parts using Geometric Features and Material Cues

by

Austin O. Myers

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2016

Advisory Committee:
Professor Yiannis Aloimonos, Chair/Advisor
Dr. Cornelia Fermüller, Co-Advisor
Professor Hal Daumé III
Professor David Jacobs
Professor John Baras

# Acknowledgments

This thesis would not have been possible without the generous guidance and support of my advisors: Prof. Yiannis Aloimonos and Dr. Cornelia Fermuller. Yiannis and Cornelia provided valuable direction and advice throughout my Ph.D while allowing me the freedom to explore many research problems. Their feedback has been instrumental in improving my ability to conduct research and clearly communicate ideas in papers and presentations. I was also fortunate enough to spend a summer at Google Research working on exciting semantic segmentation problems with Kevin Murphy, Nick Johnston, Vivek Rathod, and the rest of the VALE team. I learned invaluable lessons from many talented researchers in the Machine Perception group and I owe them a great debt of gratitude for their support and inspiration.

I would like to thank my colleagues; Ching Lik Teo, Francisco Barranco, Alex Ecins, Yezhou Yang, and many others for their insightful discussions and encouragements. I am thankful for all of those who were involved with the Computer Vision Student Seminars and the camaraderie and community atmosphere which the seminars fostered. My graduate student life would also not have gone smoothly without the help of Janice, Jenny, Fatima, Brenda, Sharron and Arlene, who have always been understanding and quick to help with any issues that arose. I would also like to thank all of the UMIACS support staff who have been outstanding in their responsiveness to the many technical issues and hard drive failures that I experienced throughout the years.

Finally, I cannot express how grateful I am for the love and support from my family, and especially from Angjoo. Thank you all so much, and to all those that I could not list here, this thesis is dedicated to you all.

# Table of Contents

# List of Figures

# Chapter 1: Introduction

Visual perception is a key ability for systems which interact with the world, and while seemingly effortless to humans, it has proven exceedingly difficult for machines. The potential importance of vision for machines has fueled decades of study in computer vision, robotics, and machine learning. With new advances in sensors, computing power, availability of data, and learning techniques, the progress in computer vision has been astounding. Recently, vision systems have quickly approached and even beaten human performance on tasks such as street sign recognition [1], face recognition [2], and image classification [3]. Still, while many efforts focus on leveraging large internet image databases, there remains a gap in what state-of-the-art methods provide and what robots need to interact with the world. Namely, in order for robots to collaborate with humans in everyday environments, they need to understand the functionality of objects and their parts.

Imagine Baxter in a kitchen, trying to serve soup from a pot into a bowl. Baxter needs to find a ladle, grab the handle, dip the bowl of the ladle into the pot, and transfer the soup into the serving bowl. Today, computer vision may allow Baxter to recognize the objects, providing a bounding box around the ladle. However, in these situations Baxter needs to not just detect the ladle, but more

importantly he needs to know which part of the ladle he can grasp and which part can contain the soup. As Gibson remarked, "If you know what can be done with a[n] object, what it can be used for, you can call it whatever you please" [4]. In this thesis, we address the novel problem of localizing and identifying part *affordance* from RGB-D data, so that a robot can explain how an object and its parts can be used, and generalize this knowledge to novel scenarios.

Gibson defined affordances as the latent "action possibilities" available to an agent, given their capabilities and the environment [4]. In this sense, for a human adult, stairs afford climbing, an apple affords eating, and a knife affords the cutting of another object. The latter is most relevant to a robot using tools in a kitchen or workshop, and we use the term *effective affordance* in this thesis to differentiate the affordances of tools from those found in other settings. We define objects with effective affordances as those that an agent can use as tools to produce an effect on another object. Man-made tools are typically composed of parts, where each part has multiple effective affordances such as cut, pound, scoop, or contain. If robots could identify these affordances, it would open the possibility to use a wide variety of tools, including those that have not been seen before.

From a computer vision perspective, predicting effective affordances from images presents many challenges. For example, tools from different categories, with unique shapes and appearances, can have parts with the same effective affordances. Furthermore, robots observe tools in a much wider variety of conditions than often found in internet images, and require precise predictions in order to interact with parts and their surfaces.

In order to address these challenges, in Chapter 2 we begin by defining the novel problem of predicting the *effective affordance* of tool parts. We then present a superpixel region based method for localizing and identifying part affordance, using geometric features learned from RGB-D data, so that robots can understand how objects and their parts can be used. In order to evaluate the proposed approach, we introduce a new RGB-D Part Affordance Dataset which consists of 105 kitchen, workshop, and garden tools. The dataset provides hand-labeled ground truth at the pixel level for more than 10,000 RGB-D images. We hypothesize that there is a deep relationship between a part's effective affordance and its geometric properties, so we analyze the performance of different feature types on our task. We show results of our method for predicting affordance of objects from a known category, and the more difficult task of predicting affordance of parts of objects from novel categories. We demonstrate that, unlike the case of RGB-D object detection, geometric features are paramount for accurate affordance identification.

In Chapter 3, we investigate the problem in a more realistic environment with object clutter and ranked affordances for each tool part. For this we propose real time solutions using two state-of-the-art methods, one with structured random forests and the other with convolutional neural networks on geometric features. Here the affordances are solved in a per-pixel manner instead of superpixels, making the approach efficient and parallelizable.

In Chapter 4, we hypothesize that in addition to geometric cues, materials properties contain important information about part affordances. However, naively combining material with geometry cues does not necessarily improve affordance

prediction, since some materials are shared in multiple affordances. So we propose a two-stream deep convolutional neural network, with one stream for predicting affordance from geometric features, a second stream trained to recognize materials from RGB data, and a final stage to combine these high level cues. We show that this approach helps affordance prediction, particularly for metallic or dark objects where geometric information is unreliable.

# Chapter 2: Predicting the Affordance of Tool Parts in RGB-D Images

## 2.1 Introduction

Every day, we use tools for ordinary activities, like cutting an apple, hammering a nail, or watering a flower. While interacting with the world, we effortlessly draw on our understanding of the functions that tools and their parts provide. Using vision alone, we can identify potential functions of object parts, in order to find a tool suited to our needs. As robots like PR2, Asimo, and Baxter begin to collaborate with humans in everyday workspaces, they will also need to understand the functionality of a wide variety of tools, even when they have never seen a particular tool before. However, from a computer vision perspective, predicting affordance from an image presents a major challenge because objects from many different categories, with unique shapes and appearances, can have parts with the same effective affordance. This is a stark contrast to most previous approaches to tool use in robotics which largely rely on instance detection of 3D object models [5, 6], and are not concerned with generalizing to a variety of tool appearances.

To address this problem, we propose a method to localize and identify part

affordance, where the output is a collection of part surfaces with their predicted affordances and 3D location, which can be used by a robot for manipulation. Given an RGB-D image, we first oversegment the scene using superpixel segmentation to divide objects into small surface fragments. Then, treating each of these regions as a candidate part, and we extract a hierarchy of features learned from raw data. We hypothesize that there is a deep relationship between a part's effective affordance and its geometric properties, so we learn higher level features from low level geometric features computed from raw depth data. Using these learned features, we train linear classifiers to predict each surface fragment's affordance. Finally, we model the interactions between neighboring superpixels using a conditional random field to encourage a consistent labeling of parts.

## 2.2 Related Work

The study of affordance has a rich history in the computer vision and robotics communities. Early work sought a function-based approach to object recognition for 3D CAD models of objects like chairs [7]. More recently, many works have focused on predicting grasping points for objects from 2D images [8] [9] [10]. For tool use, Kemp et al. [11] detects tips of tools being held by a robot. From the computer vision community, Kjellström et al. [12] classify human hand actions in context of the objects being used, and Grabner et al. [13] detect surfaces for sitting from 3D data.

Affordances might be considered a subset of object attributes, which have been

shown to be powerful for object recognition tasks as well as transferring knowledge to new categories. Ferrari and Zisserman [14] learn color and 2D shape patterns to recognize the attributes in novel images. Parikh and Grauman [15] show that relative attributes can be used to rank images relative to one another, and Lampert et al. [16] and Yu et al. [17] show that attributes can be used to transfer knowledge to novel object categories.

In the robotics community, the authors of [18] identify color, shape, material, and name attributes of objects selected in a bounding box from RGB-D data. Hermans et al. [19] explored, using active manipulation of different objects, the influence of the shape, material and weight in predicting good pushable locations. Aldoma et al. [20] used a full 3D mesh model to learn so-called 0-ordered affordances that depend on object poses and relative geometry. Koppula et al. [21] view affordance of objects as a function of interactions, and jointly model both object interactions and activities via a Markov Random Field using 3D geometric relations ('on top', 'below' etc.) between the tracked human and object as features.

Recently, unsupervised feature learning approaches have been applied to problems with 3D information. Bo et al. [22] propose hierarchical matching pursuit (HMP), and Socher et al. [23] propose using a convolutional recursive neural network to recognize objects from RGB-D images. We build upon the recent work of [24] which uses multipath HMP to achieve state of the art performance on challenging computer vision image datasets. However, in contrast to these previous works which perform whole image classification using spatial pyramids, we extract features and make predictions at the level of superpixels.

## 2.3 Part Candidates via RGB-D Superpixel Segmentation

Although 2D image segmentation is a challenging problem in computer vision, recent work has shown that incorporating depth data produces more coherent boundaries that adhere to depth discontinuities not apparent in color images [25] [26]. Usually, the goal is that each segment to correspond to an object, and great care must be taken to find segments that have not over-segmented or under-segmented an object of interest [27]. However, in our approach we consider tools composed of several parts, each formed by a collection of surfaces, so in this case oversegmentation is advantageous. Therefore, we employ an extension of Simple Linear Iterative Clustering (SLIC) for RGB-D images, since SLIC produces segments that adhere well to part boundaries, are consistent in size and shape, and are simple and fast to compute.

SLIC [28] consists of three main steps: initialization of cluster centers by sampling the image on a regular grid, performing spatially constrained $k$-means clustering on 2D spatial and color features for a fixed number of iterations, and finally post processing to enforce cluster connectivity. SLIC is first initialized with $k$ cluster centers $C_i = [x_i, y_i, l_i, a_i, b_i]^T$ sampled from the image on a regular grid. Here, $x_i$ and $y_i$ are the location of the sampled pixel in image coordinates, and $l_i, a_i, b_i$ are its CIELAB color space components. For a fixed number of iterations, pixels are assigned to clusters within a constrained spatial range using $k$-means with a custom distance metric. In [28], the distance is defined as $D = \sqrt{\left(\frac{d_c}{\alpha}\right)^2 + \left(\frac{d_s}{\beta}\right)^2}$, where $d_c$ and $d_s$ are euclidean distances between the color and spatial dimensions

respectively, and $\alpha$ and $\beta$ are hyperparameters which control how the algorithm balances color and spatial components.

We adapt SLIC for RGB-D images by clustering RGB-D pixels represented by $f = (x, c, n), f \in R^9$, where $x$ is the 3D position of a pixel, $c$ is its CIELAB color components, and $n$ is its normal vector. We define the distance metric for clustering as,

$$D = \sqrt{(d_c/\alpha)^2 + (d_s/\beta)^2 + (d_n/\gamma)^2} \tag{2.1}$$

where $d_c$ and $d_s$ are the euclidean distance between color and spatial feature vectors, and $d_n$ is the angle between normals. Hyper-parameters $\alpha, \beta$, and $\gamma$ control how the distance metric and resulting superpixels balance color, spatial, and normal components. Affordance parts are usually connected to other parts with different affordances, and these parts may only differ in either color, depth, or normal orientation. For this reason, all three components are important to achieving a good superpixel segmentation.

## 2.4 Learning Geometric Feature Representations

Given a superpixel segmentation where each segment is potentially a tool part, we next need to extract relevant information about the segment which can be used to predict its affordance. We hypothesize that affordances are strongly characterized by their geometric properties, since geometry is closely related to the physics of how objects can interact with their environment. For instance, the concavity of the inner

surface of a bowl is what allows it to have the function to contain liquid, and a blade can be used for cutting because it is long, thin, and sharp. Therefore, we make use of three geometric features depth, normal, and curvature obtained from a Kinect depth sensor to predict the affordance of a particular surface. Rather than hand engineering descriptors for each of these "raw" geometric features, we propose to learn higher level features from the data, and analyze which geometric features are most useful for affordance recognition.

Recently, unsupervised feature learning methods such as deep belief networks [23] and hierarchical sparse coding [24] have been successful in a number of vision and robotics tasks. These methods have the advantage to learn discriminative structures directly from data and encode mid-level information that are shown to be useful for various recognition tasks. Following their success, we build on the recent work of multipath hierarchical matching pursuit (M-HMP) by Bo et al [24] to learn features representations from raw features obtained from a RGB-D image.

HMP [22] is a hierarchical sparse coding method that learns feature hierarchies called *paths*. A path has a unique architecture which captures information at varying scales and abstractions, where in each layer of the hierarchy the input is encoded by sparse coding and undergoes a max-pooling operation. Specifically, we learn a hierarchy of dictionaries $D$ such that the data $Y$ can be represented by a sparse linear combination of dictionary entries.

$$\min_{D,X} \|Y - DX\|_F^2$$
$$s.t.\ \forall m, \|d_m\|_2 = 1 \text{ and } \forall n, \|x_n\|_0 \leq K \tag{2.2}$$

where $\|\cdot\|_F$ and $\|\cdot\|_0$ denote the Frobenius norm and $L_0$ norm respectively, $K$ is the sparsity regularization parameter, and $X$ is the matrix of coefficients. Given a learned dictionary, an image is represented by its coefficients or sparse codes.

In previous works, on image attribute recognition [18] or image classification [24], these codes were max-pooled over the whole image or over an image pyramid. However, we max-pool HMP features within each superpixel, which yields a feature vector for each surface. These features can be classified with a linear SVM, thereby providing a prediction of each affordance for each segment. In our experiments we use features from two-layer and three-layer architectures, which capture features at different scales and abstractions.

M-HMP is suitable for our purposes for multiple reasons. First is its strong performance in wide-array of computer vision tasks. Second, affordance can be captured by it's deep architecture allows the representation to be invariant to small local deformations and noise. Third, extracting features from multiple path allows us to capture both local and global information which is important for affordance. For example, at a local scale, a flat surface on a knife blade and a spatula may have similar geometric properties, but at a larger scale, their neighborhood structure is not the same. Furthermore, the advantage of using a feature learning approach as opposed to using a collection of hand-engineered features is that it provides a consistent way to extract features from these raw data. This provides an equal footing

for comparing which raw features are discriminative for affordances recognition.

## 2.5 Affordance Prediction Refinement

Given an RGB-D image we compute superpixel segmentation using the method in Section 2.3. We predict the affordance of each super pixel with a Conditional Random Field (CRF) [29]. Regarding each superpixel segments as a random variable $\vec{S} = \{s_1, \ldots, s_n\}$, we define a graph $G(S, E)$, with label assignments $\vec{c} = \{c_1, \ldots, c_n\}$ for each superpixel. We model the posterior distribution of each image as

$$-\log P(c|G) = \sum_{s_i \in S} \Phi(c_i|s_i) + w \sum_{(s_i,s_j)\in E} \Psi(c_i, c_j|s_i, s_j), \tag{2.3}$$

where $\Phi$ is the unary potential, $\Psi$ is the edge potential that smoothes label discontinuities depending on data, and $w$ is a weight parameter. For the unary potential, we first train a multi-class linear SVM using spatially max-pooled M-HMP features extracted from superpixel segments. We use the output of the SVM directly for the unary potential, $\Phi(c_i|s_i) = -\log P(c_i|s_i)$. The pairwise potential $\Psi$ is defined as

$$\Psi(c_i, c_j|s_i, s_j) = \left(\frac{B(s_i, s_j)}{1 + ||s_i - s_j||}\right) \delta(c_i \neq c_j), \tag{2.4}$$

where $\delta$ is an indicator function, and $B(s_i, s_j)$ is a regularization term defined to be the length of the shared boundary between $s_i$ and $s_j$ [30]. Since we only have one parameter $w$, we set the value by cross validation on the training data. We initialize the graph using the unary potentials and solve for the optimal labeling $c^\star$ using the alpha-expansion graph-cut algorithm of [31]. The proposed S-HMP framework is illustrated in Figure 2.1.

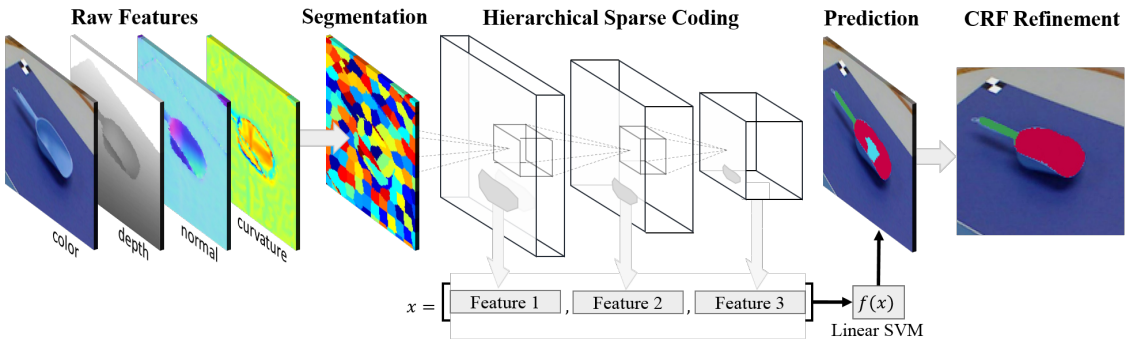



Figure 2.1: Affordance detection using S-HMP. An RGB-D image is segmented into superpixels, where each segment serves as a candidate part surface (left). For each superpixel, hierarchical sparse codes are extracted from geometric features such as depth, surface normal, and curvature information (middle). Superpixels' codes are pooled and then classified using a linear SVM, and the final labeling is refined using a CRF (right).

## 2.6 RGB-D Part Affordance Dataset

To investigate the problem of localizing and identifying affordance, we propose a new RGB-D Part Affordance Dataset which focuses on everyday tools annotated with the affordances of their parts. We consider tool *parts* corresponding to a collection of *surfaces* which provide a certain functionality. We define each surface's effective affordances by the way it comes in contact with the objects that they affect. For example, a coffee mug has two affordance parts, the inner surface and the outer surface. The inner surface of a mug has the effective affordance "contain", because it comes in contact with the liquid that is contained. The surface of the mug's handle has the affordance "grasp" as it can be tightly held by a hand or robot gripper. The dataset provides pixel-level affordance labels for 105 kitchen, workshop, and garden tools. The tools were collected from 17 different categories covering seven types

| Affordance | Description |
|---|---|
| `grasp` | Can be enclosed by a hand for manipulation (handles). |
| `cut` | Used for separating another object (the blade of a knife). |
| `scoop` | A curved surface with a mouth for gathering soft material (trowels). |
| `contain` | With deep cavities to hold liquid (the inside of bowls). |
| `pound` | Used for striking other objects.(the head of a hammer). |
| `support` | Flat parts that can hold loose material (turners/spatulas). |
| `wrap-grasp` | Can be held with the hand and palm (the outside of a cup). |

Table 2.1: Description of the seven affordance labels.

of affordance which are summarized in Table 2.1. Each affordance is represented by objects from a variety of categories with different appearances, as shown in Figure 2.2. For example, parts with the affordance "cut" are found in kitchen knives, workshop saws, and garden shears.

While there are several RGB-D object datasets, most are designed for instance and category level object recognition [32], attribute learning [18] or for specific robotic gripping locations [33]. In addition to testing with tools from a known category, the dataset is designed for evaluating part affordance identification for objects from completely novel categories. To our knowledge this is the first dataset specifically designed for robots to identify and localize part affordances from RGB-D data.

Data was collected using a Kinect sensor, which records RGB and depth images at a resolution of 640×480 pixels. Since many of the parts we want to capture are small, we collected data at the minimum distance required for accurate depth

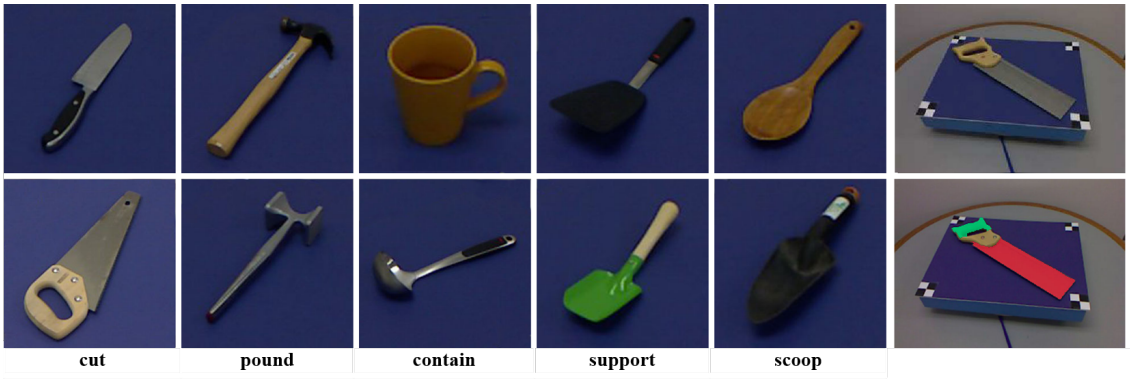



Figure 2.2: Sample objects from the RGB-D Part Affordance Dataset. (Left) Each column shows two close-up examples of objects with parts that share the same affordance. (Right) An example of a full frame image (top) with hand-labeled ground truth (bottom).

readings, approximately 0.8 meters. We recorded each tool on a revolving turntable to collect images covering a full 360° range of views. On average, approximately 300 frames are captured for each tool, producing more than 30,000 RGB-D image pairs. Of these, more than 10,000 images have pixel-level ground truth affordance labels.

Labeling such a large dataset at the pixel level would ordinarily be a very expensive task, so we reduced labeling effort by taking advantage of the superpixel segmentation method described in Section 2.3. We segmented images into superpixels, at a finer scale than in our experiments, and allowed annotators to click on the segments that corresponded to the desired affordances. Since data was also collected on a rotating turntable, after an annotator finished labeling a frame, we propagated labels to the next image based on superpixel proximity and similarity. This greatly reduced labeling effort, since in the majority of images annotators only needed to correct a small number of segments, and only occasionally would annotators need to click on many segments in the image. While still a time consuming task, labeling

approximately 10,000 images became manageable using this approach.

## 2.7 Experiments

In our experiments, we first analyze the effectiveness of individual raw feature types in order to test our hypothesis that geometric features are essential for affordance identification. We then present the results of our method for known category and novel category affordance localization and prediction. We perform experiments using the RGB-D Part Affordance Dataset described in Section 2.6. In the known category setting, we evaluate using a 10-fold randomized leave-one-out procedure, as used in the RGB-D dataset for category-level recognition [32]. For each fold, an object from each category is left out for testing, and the remaining objects are used for training. This provides approximately 8,000 RGB-D images for training and 2,000 for testing. Performance is reported as the average class accuracy over the 10 folds. For the novel category setting, categories of the dataset are split into two groups, so that each affordance is represented by tools from different environments in training and testing. In all experiments, we measure performance using average class accuracy.

### 2.7.1 Implementation Details

For superpixel segmentation, we found that parameters $\alpha = 1$, $\beta = 0.05$, and $\gamma = 0.125$ produce segments that adhere well to object boundaries. On average this produces 300 superpixel segments for each image. To compute M-HMP features and

dictionaries we build on the publicly available M-HMP code of Bo et al [24]. We learn dictionaries for five raw feature types; RGB color, grayscale, depth, surface normal, and curvature. Surface normals and curvature images are computed from depth images. For each raw feature, we learn two paths $P1$ and $P2$. $P1$ is a two layer architecture, which learns dictionaries on $5 \times 5$ raw feature patches in the first layer, followed by $4 \times 4$ max-pooling. The second layer learns dictionaries on $4 \times 4$ patches of the resulting sparse codes, corresponding to $16 \times 16$ raw feature patches. $P2$ is a three layer architecture, whose first two layers are the same as the layers of $P1$. The third layer learns dictionaries on $3 \times 3$ patches of $3 \times 3$ max-pooled sparse codes from layer 2, corresponding to $36 \times 36$ raw feature patches. To learn each feature hierarchy, we sample approximately 1,000,000 patches from raw feature images, and normalize the brightness of patches by subtracting their mean. M-HMP features are extracted for each segment and concatenated, resulting in a 4600 dimensional feature. We also ensure that test data is not used for dictionary learning by splitting the data into two groups and learning a separate set of dictionaries for each group. At test time, features are extracted using the dictionaries which did not use the test data for learning.

### 2.7.2 Comparison of Individual Features

Recent results for RGB-D object recognition have shown that visual features outperform geometry features in instance and category level recognition settings [32] [22]. However, for the task of affordance identification, we hypothesize

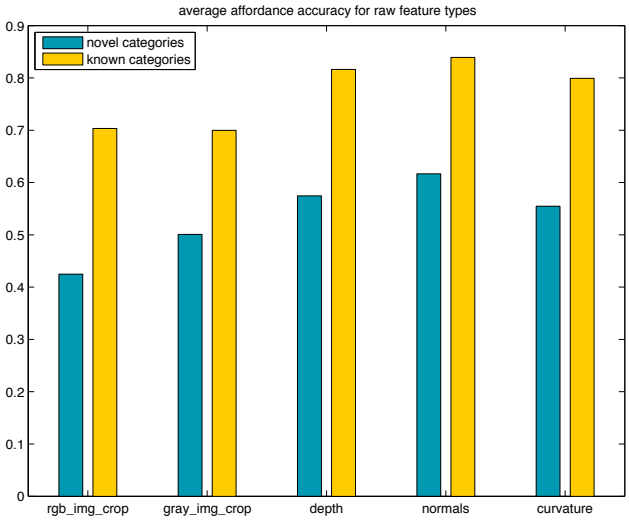



Figure 2.3: Average affordance accuracy for each of the raw features RGB, grayscale, depth, normals, and curvature. Results are shown for known category and novel category settings.

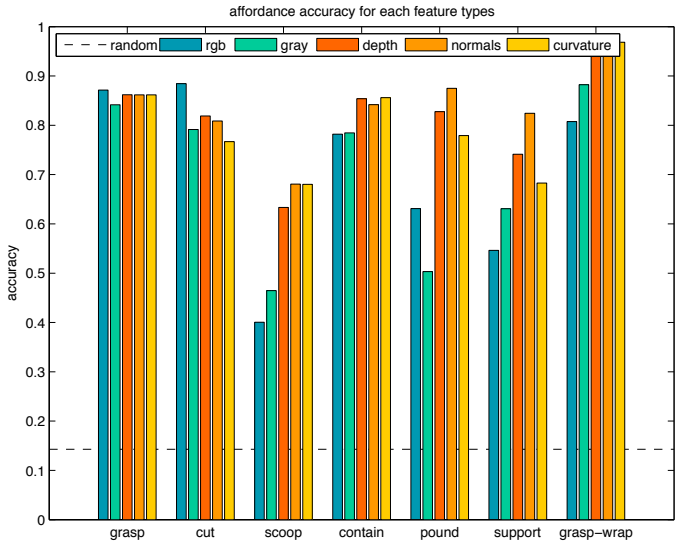



Figure 2.4: Comparison of different raw features for each affordance in the known category setting. Blue-green bars show the results of "Appearance" features (rgb and grayscale). Orange-yellow bars show results of the "Geometry" features (depth, normal, curvature).

that geometric features are essential. To validate this hypothesis, we train an independent multi-class linear SVM for each of the five raw features and compare their performance, as shown in Figure 2.3. In both known and novel category settings, we find that geometric features significantly improve over appearance features. We can also see that the novel category setting is significantly more challenging than the known category setting as expected, since novel objects have very different color and shape appearance. For a more detailed analysis, we also show the accuracy of each affordance separately, in the known category setting, in Figure 2.4. From this breakdown, we can see that geometry features are superior for "scoop", "contain", "pound", "support", and "wrap-grasp". All five feature types show similar performance for "grasp" and "cut", with the exception of RGB outperforming other features for "cut". We suspect that RGB information is useful for cut since many of the cutting blades lay flat against the table, with few discriminative geometric features. Geometric features offer substantial improvement for other affordances. Based on these experiments, we find grayscale, depth, normal, and curvature to be the most effective features. We concatenate these M-HMP features and refer to this final feature representation as *All*. We also refer to the concatenation of RGB and grayscale as *Appearance*, and the concatenation of depth, normal, and curvature as *Geometry*.

### 2.7.3 Results for Known and Novel Categories

We evaluate our complete framework using *All* features with CRF refinement for affordance identification of parts from objects of known and novel categories. We compare the performance of *Appearance*, *Geometry*, *All*, and our framework in Figure 2.2. As anticipated from our initial experiments, geometry features have relative improvement over appearance features in both known and novel settings by 18% and 38% respectively. Even more telling, combining both appearance and geometric features does not provide significant improvement for either task, a significantly different conclusion from category-level object recognition results, where combining appearance and geometric features increases performance [32] [22]. This indicates that geometric features are primary for the problem of predicting affordance.

We show example results of our full framework for both known and novel category settings in Figure 2.5 (a-b). While adding CRF does not appear to change performance significantly, it is an important step for producing a final output that is useful for a robot. The refinement step removes small inconsistent regions so that the output is more consistent and usable by a robot. Overall the CRF does not have a large impact on performance, since while refinement corrects small errors, it also makes mistakes when small correct regions are surrounded by an incorrectly labeled neighborhood. In Figure 2.5 (c) we show results where CRF benefits the framework, as well as cases where it increases errors.

Overall, the affordances of parts are well localized, and the 3D labeled points could be used directly by a robot for tool manipulation. Novel category results are

| Known Object Category | | | |
|---|---|---|---|
| Appearance | Geometry | All | All + CRF |
| $73.2 \pm 3.5$ | $86.5 \pm 6.6$ | $86.2 \pm 5.6$ | $86.5 \pm 5.0$ |

| Novel Object Category | | | |
|---|---|---|---|
| Appearance | Geometry | All | All + CRF |
| 46.0 | 63.6 | 64.8 | 64.8 |

Table 2.2: Part affordance identification on the RGB-D Part Affordance Dataset for known and novel object settings. Performance using appearance features, geometry features, all features, and the complete framework.

also promising, for example the blade of the "pruning shears" is correctly labeled as "cut" even though the system only saw "knives" and "scissors" during training. However, while some affordances are accurately identified, others are easily confused as illustrated by example errors in Figure 2.5 (b). In one case, the tips of a small pair of "scissors" are mistaken as part of the handle. In another, the flat surface of the "shovel" is correctly labeled as "support", whereas the edges are labeled "scoop". As shown by the example errors in Figure 2.5 (b-c), we found that affordances such as "contain" and "scoop" or "support" and "scoop" are sometimes confused. This occurs for example, when one trowel's head, which is flatter than others, is often confused as a supporting part. In some views, food scoops are confused with containing parts, like the concavity of a bowl. These mistakes are not completely wrong, and would actually be quite useful for a robot. More importantly, these mistakes reveal the similarity shared between related affordances, opening new avenues for future

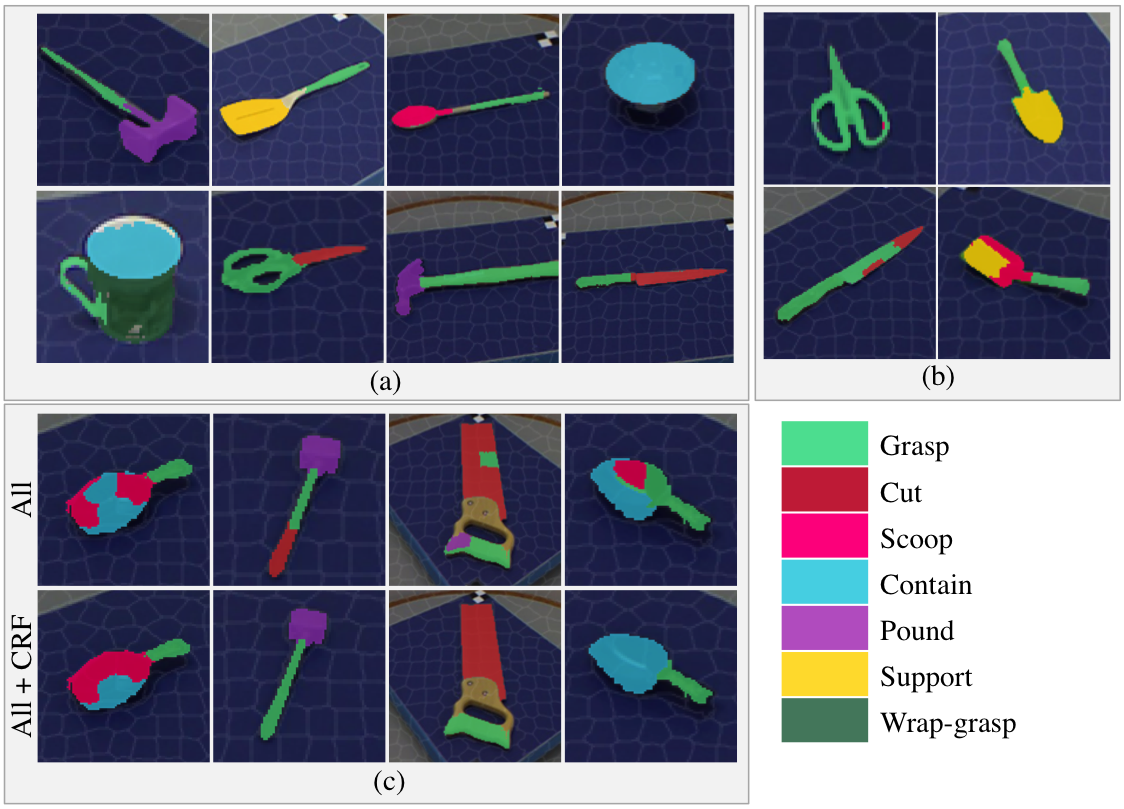



Figure 2.5: Examples of results from our method from known category (a. top row) and novel category (a. bottom row) experimental settings. Failure cases show missed part errors and confusion between similar affordances (b). CRF refinement is an important step for producing output that is consistent and usable for a robot (c). Although CRF improves labeling in many cases (c. columns 1-3), it can also make additional mistakes (c. column 4).

work.

## 2.8 Conclusion

For robots to collaborate with humans in everyday workspaces, it is critical that they understand the functionality of objects and their parts. However, recognition of part affordances is a challenging problem due to the fact that parts with

similar affordances can come from a variety of object classes.

In this chapter, we have presented an approach to localize and identify the affordance of tool parts, and we introduced a new RGB-D Part Affordance Dataset which we used to evaluate our approach. Through experiments, we also demonstrated the importance of geometry for affordance prediction, showing that geometric features outperform appearance features by a significant margin, and that the addition of appearance features does not yield a significant performance improvement. Finally, we also showed that affordances can be learned from a set of training objects, and successfully applied to novel objects from classes not seen in training.

This work shows the potential for learning to detect affordances, as apposed to previous techniques focused on detecting object instances. In contrast to works which provide image labels or bounding boxes, the proposed approach localizes and predicts affordances at the pixel level, which provides information that can be readily used in robotics applications. This also opens new research directions, by raising interesting questions about the relationships between similar affordances, and the potential for tool parts to have multiple affordances.

# Chapter 3: Dense Part Affordance Prediction in Real-time

## 3.1 Introduction

In the previous chapter we developed a framework that can localize and identify tool part affordances, using superpixel segmentation and HMP to learn a hierarchy of geometric features from raw depth data. This demonstrated the potential for learning to predict the affordance of tool parts in contrast to traditional instance based object recognition approaches commonly used in robotics applications. However, the proposed S-HMP approach is limited by the quality of the superpixel segmentation and more importantly the computational complexity of sparse coding for patches in the image. In this chapter, instead of predicting affordance of superpixel regions, we propose two approaches for dense part affordance prediction in real-time. The first approach leverages the fast inference of structured random forests (SRF) to detect part affordances in real-time. Secondly, we proposed a deep convolutional neural network approach which learns to predict part affordance from raw geometric features. Finally, we also demonstrate the robustness of the approaches in challenging real-world situations containing clutter, occlusions and viewpoint changes which were not explored in prior works.

## 3.2 Related Work

Recently, deep learning methods have achieved impressive results on visual classification tasks like street sign recognition [1], face recognition [2], and image classification [3]. These approaches have demonstrated the ability to learn complex patterns by building increasingly abstract representations layer by layer, and have been able to leverage advances in GPU hardware to learn from large amounts of data. Convolutional Neural Networks (CNNs) have subsequently been applied to many problems in computer vision very recently, but much less deep learning work has addressed RGB-D data and vision problems in robotics. Recently, [34] used CNNs for semantic segmentation of RGB-D scenes, and [23] proposed a convolutional recursive neural network to recognize objects from RGB-D images.

## 3.3 Robust Geometric Features

The key hypothesis of this thesis is that shape and geometry are physically grounded qualities which are deeply tied to the affordances of a tool part. When characterizing geometric qualities of a part, it is important that the features we compute are robust to variations, such as changes in viewpoint. At the same time, we would like to gain insight into the influence of basic geometric measures. Therefore, we leverage simple geometric features, such as surface normals and curvature, to learn the relationship between geometry and part affordance. In order to detect affordances for a variety of tools in cluttered scenes with occlusions, we derive the

following local geometric features from small RGB-D input patches.

### 3.3.1 Depth Features

We first apply smoothing and interpolation operators to reduce noise and missing depth values. Then, we remove the mean from the patch to gain robustness to absolute changes in depth. These patches are used directly by HMP to learn hierarchical sparse code dictionaries. In the first layer, HMP captures primitive structures such as depth edges at various orientations, and higher layers encode increasingly abstract representations [22]. To provide comparable depth edge information to the SRF, we compute histograms over depth gradients (HoG-Depth). Similar to the 2D Histogram of Gradients (HoG) image descriptor [35], we compute gradients on the depth image and quantize them into four orientations to create a compact histogram feature.

### 3.3.2 Surface normals (SNorm)

We use the depth camera's intrinsic parameters to recover the 3D point cloud, from which we can estimate 3D surface normals. As with the depth, we remove the patch mean during feature learning, to make the representation more robust to changes in viewpoint.

### 3.3.3 Principle curvatures (PCurv)

The principle curvatures [36] are an extrinsic invariant of the local patch geometry, and are independent of viewpoint. The principal curvatures $(\kappa_1, \kappa_2), \kappa_1 \geqslant \kappa_2$ characterize how the surface bends in different directions.

### 3.3.4 Shape-index and curvedness (SI+CV)

The shape index (SI) and curvedness (CV) measures were introduced by Koenderink et al. [37] to characterize human perception of shape. These measures, which are derived from $(\kappa_1, \kappa_2)$, are also viewpoint invariant and are defined as

$$SI = -\frac{2}{\pi}\arctan\left(\frac{\kappa_1 + \kappa_2}{\kappa_1 - \kappa_2}\right), CV = \sqrt{\frac{\kappa_1^2 + \kappa_2^2}{2}} \tag{3.1}$$

SI and CV are continuous in the range $[-1, +1]$, where the shape index captures the type of local shape (elliptic, parabolic, etc.) and the curvedness its perceived strength.

## 3.4 Structured Random Forest

The random forest (RF), introduced by [38], is an ensemble learning technique that combines $K$ decision trees, $(T_1, \cdots, T_K)$, trained over random permutations of the data to prevent overfitting. The output of the model can either be a class label (for multilabel classification) or a continuous value (for regression). The main advantage of random forests is that inference is extremely efficient [39], since data

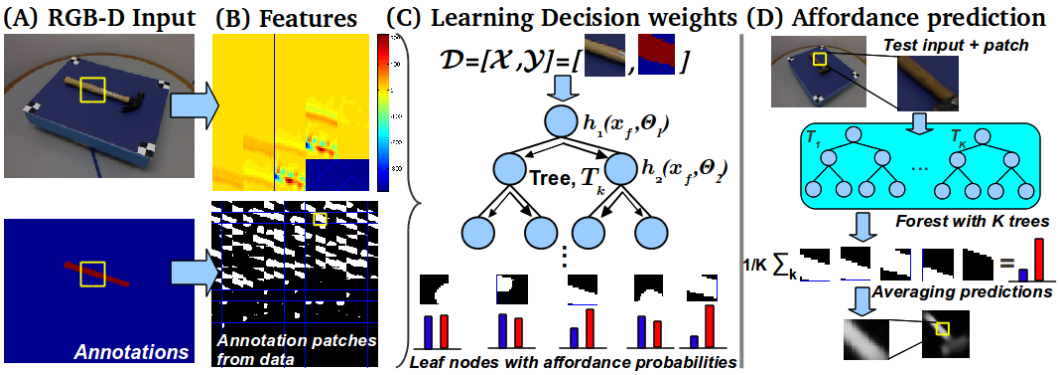



Figure 3.1: Affordance detection using SRF. (A) Input image with example patch highlighted. (B) Features extracted from each patch (top) and sampled annotation patches from data (below). (C) Training different patches, $\mathcal{X}$ with corresponding binary affordance annotations, $\mathcal{Y}$, learns the optimal $\theta_j$ at each split node. The leaf nodes store per pixel confidence scores for each $\mathcal{Y}$ encountered. (D) During inference, a test patch is assigned to a leaf node that contains affordance prediction. Averaging the predictions over the $K$ trees produces an affordance confidence score per pixel.

only needs to be passed through several binary decision functions. Due to their speed and flexibility, RFs have been widely applied in both computer vision and robotics problems.

We propose an approach using a *structured* random forest (SRF), an extension of the standard RF that imposes structured constraints on the input and output. This enables the SRF to learn more expressive information, such as shapes, sizes, or even abstract relationships between entities while still retaining all the inherent advantages of standard RFs. Structured random forests were introduced by [40] to impose spatial constraints for scene segmentation, and they were recently extended by [41] for 2D edge detection.

Different from these previous works, we impose here a novel structure that

relates affordances to the local patch geometry and shape. To this end, we train a SRF that takes as input $\mathcal{X}$, features from local $N \times N$ patches described in Section 3.3 with pixel accurate annotations of the target affordance, $\mathcal{Y}$ (Figure 3.1 (B)). The annotations impose the expected spatial structure of how the affordance should appear in the final prediction. For the $j^{\text{th}}$ split (internal) node, we train a binary decision function $h(x, \theta_j) \in \{0, 1\}$ over random subsets, $x \in \mathcal{X}$, of the input features so that the parameters $\theta_j = (f, \rho)$ send $x(f)$ (where $f$ is the feature dimension for each feature described in Section 3.3) to the left child when $h(\cdot) = 1$ if $[x(f) < \rho]$ and to the right child otherwise. The decision threshold, $\rho$, is obtained by maximizing a standard information gain criterion $G_j$ over $\mathcal{D}_j \subset \mathcal{X} \times \mathcal{Y}$, the features and annotations:

$$G_j = H(\mathcal{D}_j) - \sum_{c \in \{L,R\}} \frac{|\mathcal{D}_j^c|}{|\mathcal{D}_j|} H(\mathcal{D}_j^c) \tag{3.2}$$

where $D_j^c, c \in \{L, R\}$ indicates the portion of the data that is split by $\rho$ into the left and right child nodes respectively. We use here the Gini impurity measure: $H(\mathcal{D}_j) = \sum_y p_y(1 - p_y)$ with $p_y$ denoting the proportion of features in $\mathcal{D}_j$ with ownership label $y \in \mathcal{Y}$. Eq. (3.2) is computed via an intermediate mapping $\Pi : \mathcal{Y} \mapsto \mathcal{L}$ of structured affordance labels into discrete labels $l \in \mathcal{L}$ following [41]. To determine $\Pi$, we first cluster via k-means random annotation patches that have the same affordance labels and select the largest $|\mathcal{L}|$ cluster centers. We repeat the training procedure until a maximum tree depth, $D_t$, is reached and we store at the leaf nodes per pixel confidence scores for each affordance annotation patch

encountered during training. (Figure 3.1 (C)). Each tree in the SRF therefore learns jointly, the 2D spatial structure *together* with the 2.5D features that describe the affordance within a patch. Inference using the trained SRF is extremely simple. Given a forest of $K$ trees and a testing patch with extracted features, the learned decision thresholds in each split node will send the patch to a leaf node that contains the predicted affordance labeling and confidence scores. We then average all $K$ predictions for the final prediction (Figure 3.1 (D)).

In our implementation, we train a SRF with $K = 8$ trees with a maximum training depth of $D_t = 64$. We use patches of size $N = 16$ and we set $|\mathcal{L}| = 10$ cluster centers for $\Pi$. Training over the entire Affordance RGB-D dataset (Section 3.6.2) in parallel with approximately 5000 RGB-D images per split takes around 20 minutes on a 16 core Xeon 2.9GHz machine with 128GB of ram. Inference for a single RGB-D image of size $(640 \times 480)$ (height, width), takes an average of 0.1s which includes the time for feature extraction.

## 3.5 Convolutional Neural Networks

In Chapter 2, we used hierarchical matching pursuit (HMP) to compute hierarchical sparse codes from raw geometric features, with the goal of learning multiple layers of features with each successive layer encoding increasingly abstract representations. Unfortunately, HMP is computationally expensive, since for every patch we need to compute sparse coefficients using orthogonal matching pursuit (OMP), so it is not well suited for real-time robotics applications. While SRFs are efficient

enough to run in real-time scenarios, they compute binary decision functions from existing features, and do not learn new feature representations from the data. So, in order to learn features for predicting affordance in real-time, we we propose a Deep Convolutional Neural Network (DCNN) based approach.

Convolutional Neural Networks (CNNs) have recently reemerged in the in the computer vision community, obtaining state-of-the-art results on a number of tasks. Like HMP, CNNs learn multiple layers of feature representations, with each layer learning higher-level and more abstract features. Whereas HMP uses a learned dictionary to compute sparse codes at each layer, CNNs convolve learned filters over the input features to produce the layer's output, before transforming the outputs using a non-linear activation function. In addition, HMP and CNNs both use max-pooling layers to aggregate features over image patches and to achieve invariance to small spatial perturbations. In practice, CNNs have also become more accessible with the recent development of publicly available libraries like Caffe [42]. Since CNNs are feedforward and highly parallelizable, they can also be run efficiently on GPUs, and with appropriate architectural considerations can enabling real-time operation.

Following the HMP architecture used in Chapter 2, we propose a DCNN architecture with 3 intermediate convolution layers, where the outputs of each convolution layer are passed through a ReLU non-linearity. The first two convolution layers are also followed by a max-pooling operation on overlapping $3 \times 3$ patches. After 3 convolution layers, we apply a final pair of convolution and logistic layers to output an affordance probability map. We note that the logistic classifiers are independent,

| Name | Kernel size (dilation) | Stride | Output size ($C \times H \times W$) |
|---|---|---|---|
| `conv_1` | $7 \times 7$ | 2 | $64 \times 105 \times 140$ |
| `pool_1` | $3 \times 3$ | 2 | $64 \times 52 \times 70$ |
| `conv_2` | $5 \times 5$ | 1 | $128 \times 52 \times 70$ |
| `pool_2` | $3 \times 3$ | 1 | $128 \times 50 \times 68$ |
| `conv_3` | $3 \times 3$ (2) | 1 | $256 \times 50 \times 68$ |
| `conv_class` | $1 \times 1$ | 1 | $7 \times 50 \times 68$ |
| `sigmoid` | | | $7 \times 50 \times 68$ |

Table 3.1: Affordance CNN Architecture.

which in contrast to a softmax layer, allows for multiple positive classes.

We also process patches with a stride of 2 pixels in the first convolution and pooling layers, while all remaining layers use a stride of 1. Many similar architectures also use a stride of 2 in the later convolution and pooling layers [3, 43], however this would further reduce the spatial resolution of the output probability map. Instead, we achieve equivalent receptive fields while preserving spatial resolution by using dilated convolution kernels in later layers. The overall architecture is detailed in Table 3.1.

In order to train the network using the different geometric features described in Section 3.3, we concatenate the desired feature maps into a tensor which is then input to the first convolution layer. For each of the features, we center and normalize the data using each channel's mean and standard deviation, computed on all pixels in the training data.

## 3.6 Experiments

We conduct several experiments to compare the proposed approaches with related approaches in the literature, understand the importance of different geometric features, and evaluate the performance of the proposed methods on everyday tools in controlled and cluttered environments. First, in Section 3.6.1, we present experiments on a baseline task of predicting good robotic grasping regions (a specific affordance), and compare the proposed S-HMP and SRF approaches with the approach of Lenz et al. [33]. In Section 3.6.2, we describe extensions made to the part affordance dataset, which is used to evaluate the S-HMP, SRF, and CNN approaches. We present the evaluation metrics used for all experiments in Section 3.6.3. Lastly, we present and discuss the results of our experiments in Sections 3.6.4-3.6.6.

### 3.6.1 Grasping Detection Baseline

Since there are no prior works on part affordance detection with which we can compare, we turn to the related robotics task of determining where to grasp (a specific affordance) as a baseline. We used the recently introduced Cornell Grasping Dataset of Lenz et al. [33] and compare against their sparse auto-encoder based approach to validate the effectiveness of our SRF and S-HMP approaches. The dataset contains 1035 RGB-D images of 280 graspable objects, where objects are captured from a small discrete number of viewpoints. Each image contains a single object, and is annotated with a set of rectangles indicating good or bad graspable locations. Following the testing procedure in [33], we averaged results from 5 random

splits, and report both recognition accuracy and detection accuracy. For detection, we report the point-wise metric following [33] and [8], which considers the detection a success if it is within some distance from at least one ground-truth rectangle center. In order to use S-HMP in this setting, we treat the candidate rectangles as superpixel segments, and perform max-pooling over the rectangle to make a prediction. To obtain structured labels for the SRF, we estimated the ground-truth annotations of graspable regions by first applying a mask obtained over all graspable rectangles followed by a edge detection and hole filling operation (Figure 3.2).

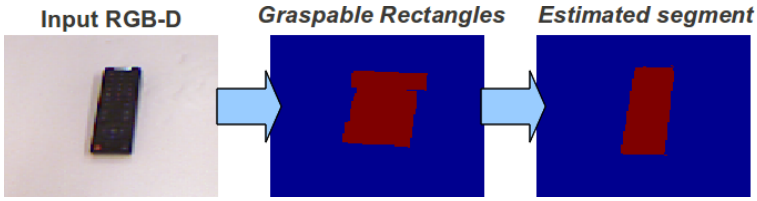



Figure 3.2: Estimating pixel accurate annotations from the Cornell Grasping Dataset. (Left) Input RGB image. (Middle) Overlay of several graspable rectangles. (Right) Edge detection and hole filling produces a pixel accurate segment.

We applied the proposed approaches to the Cornell Grasping Dataset and compared recognition and detection results to those of the sparse auto-encoder (SAE) with a two-stage structured regularization in [33]. Table 3.2 summarizes the recognition accuracy, $r_a$, and detection accuracy (point-wise), $d_a$, of the SRF, SAE, and S-HMP methods. In order to highlight the contribution of the structured constraints in the SRF, we trained a standard random forest (RF) with 20 trees over the annotated grasping rectangles in the dataset, using the *same* feature set of the SAE: RGB + Depth + SNorms.

We note first that using the baseline feature set used in SAE with a standard RF results only in mediocre performance. By adding the structured constraints and the proposed robust features, the SRF is able to achieve recognition and detection performances comparable to the deep learning based SAE. S-HMP outperforms the other approaches by a large margin, achieving state-of-the-art performance for this dataset. It is important to note however, that the SRF provides very reasonable predictions of graspable locations with pixel-wise accuracy (Figure 3.3), within a *fraction* of the time needed for inference using SAE (30s) vs. 0.1s in SRF. Such real-time performance is crucial for practical robotics applications and we show in the supplementary video an example of real-time detection over the cluttered RGB-D Affordance Dataset.

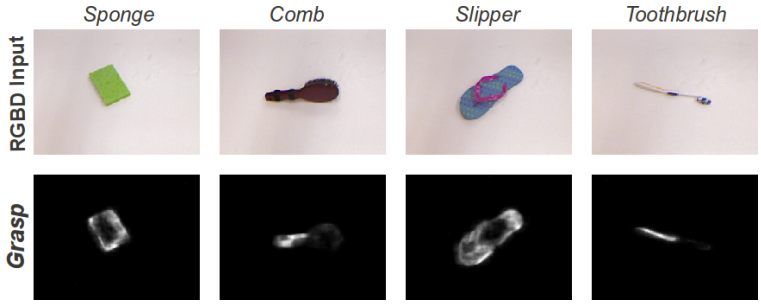



Figure 3.3: Grasping locations predicted by SRF. (Top) Input RGB-D images for four example objects. (Bottom) Predicted graspable locations. Notice the large difference in shape of the graspable regions. Brighter means higher probability.

### 3.6.2 Dataset

In Section 2.6 we introduced a new RGB-D Part Affordance Dataset to investigate the problem of localizing and identifying part affordances. The dataset

| Method | $r_a$ % | $d_a$ % |
|---|---|---|
| RF | 85.3 | 62.5 |
| SRF | 93.5 | 87.0 |
| SAE [33] | 93.7 | 88.4 |
| S-HMP | **95.2** | **92.0** |

Table 3.2: Performance comparisons in the Cornell Grasping dataset.

provides pixel-level affordance labels for 105 kitchen, workshop, and garden tools from 17 different categories. However, results from experiments on the dataset showed that some affordance predictions were confused with similar affordances. Although these labelings do not match the ground truth, in some cases they are not completely wrong. For example, the inside of a bowl might be annotated with the affordance "contain", but a prediction of scoop is somewhat reasonable. This highlights the fact that tool parts may have *multiple* affordances, where some affordances may be more likely than others. To address this, we engaged several human annotators to *rank* affordances for each part with respect to the essential affordance category, while allowing for ties. This allows us to determine, on an ordinal scale, how well the affordance detector generalizes to related affordances which is important when novel objects are observed. In addition, we expanded the dataset with three sequences of approximately 1000 RGB-D frames, each collected by a mobile robot observing novel tools in clutter under changing viewpoints. Example frames are shown in Figure 3.4 (left).

### 3.6.3 Evaluation Metrics

We use three evaluation metrics to provide different perspectives on the performance of our approaches over the RGB-D Part Affordance dataset. The proposed approaches output a probability map over the image for each affordance, which can be evaluated against ground truth labels to fairly compare their performance. First, we use the *Weighted F-Measure*, $F_\beta^w$, introduced recently by Margolin et al. [44] to evaluate saliency maps with continuous valued responses against binary valued ground-truths.

The well-known F-measure $F_\beta$ is defined by the harmonic mean of the precision and recall values: $F_\beta = (1+\beta^2).\frac{Pr.Rc}{\beta^2.Pr+Rc}$ and is used as a measure of the accuracy of the $Pr$ and $Rc$ scores. The weight $\beta$ is a positive value that gives preferences to either $Rc$, when $\beta > 1$, or $Pr$, when $\beta < 1$. The *Weighted F-Measure*, is an extension of the F-measure,

$$F_\beta^w = (1+\beta^2)\frac{Pr^w.Rc^w}{\beta^2.Pr^w + Rc^w}, \text{with } \beta = 1 \tag{3.3}$$

where $Pr^w$ and $Rc^w$ are *weighted* versions of the standard precision $Pr = \frac{TP}{TP+FP}$and recall $Rc = \frac{TP}{TP+FN}$ measures. Here, $TP, TN, FP, FN$ refer to true positives, true negatives, false positives and false negatives respectively. The key insight from [44] is to extend the standard precision and recall measures with weights derived by comparing the binary ground-truth and the continuous valued responses in order to reduce biases inherent in the standard measures. To do this, the authors proposed weights that measure the dependency of foreground pixels (pixels clustered together

near the ground-truth are weighted higher), and assign lower weights to pixels far from the ground-truth.

Since the ground-truth in the RGB-D Affordance dataset provides rankings across multiple affordances, for a second measure we define a *rank* weighted $F_\beta^w$,

$$R_\beta^w = \sum_r w_r . F_\beta^w(r), \text{with} \sum_r w_r = 1 \tag{3.4}$$

that sums weighted $F_\beta^w(r)$ over their corresponding $r$ ranked affordances. The ranked weights $w_r$ are chosen so that the top ranked affordance is given the most weight, followed by the secondary affordance and so on. This allows us to capture if the detector is generalizing across multiple affordances appropriately. Note that when we impose $w_1 = 1$, (3.4) reduces to (3.3), where we consider only the top ranked affordance.

Finally, we use a third measure to evaluate whether multiple affordance predictions agree with the ground-truth rankings. We rank the continuous affordance predictions at each pixel, and compute the *ranked* correlation score, Kendall's $\tau_k \in [-1, 1]$ [45]. $\tau_k$ approaches 1 as the predicted ranks agree more closely with the ground-truth, but nears -1 as the ranks are reversed. We report $\overline{\tau}_k \in [-1, 1]$, the average $\tau_k$ of all pixels over the test images.

### 3.6.4 Comparison on RGB-D Tool Dataset

We report results that demonstrate the performance of our approaches using the proposed metrics described above: $(F_\beta^w, R_\beta^w, \overline{\tau}_k)$ for affordance detectors trained using the S-HMP, SRF, and CNN methods. We used the same train/test splits of

| Affordance | CNN $(F_\beta^w, R_\beta^w, \overline{\tau}_k)$ | HMP $(F_\beta^w, R_\beta^w, \overline{\tau}_k)$ | SRF $(F_\beta^w, R_\beta^w, \overline{\tau}_k)$ |
|---|---|---|---|
| `grasp` | 0.520, 0.216, 0.908 | 0.367, 0.149, 0.711 | 0.314, 0.133, 0.409 |
| `cut` | 0.534, 0.062, 0.916 | 0.373, 0.043, 0.831 | 0.285, 0.033, 0.798 |
| `scoop` | 0.581, 0.101, 0.854 | 0.415, 0.046, 0.627 | 0.412, 0.097, 0.559 |
| `contain` | 0.815, 0.176, 0.916 | 0.810, 0.168, 0.814 | 0.635, 0.142, 0.579 |
| `pound` | 0.686, 0.045, 0.949 | 0.643, 0.035, 0.787 | 0.429, 0.033, 0.801 |
| `support` | 0.701, 0.049, 0.911 | 0.524, 0.030, 0.717 | 0.481, 0.039, 0.724 |
| `wrap-grasp` | 0.864, 0.115, 0.963 | 0.767, 0.102, 0.867 | 0.666, 0.089, 0.821 |
| Mean | **0.672**, **0.109**, **0.917** | 0.557, 0.082, 0.751 | 0.460, 0.081, 0.643 |

Table 3.3: Performance over the RGB-D Affordance dataset.

the RGB-D Affordance dataset used for the experiments presented in Section 2.7, and we report averaged results over the splits. We used the features described in Section 3.3 for a fair comparison. Table 3.3 summarizes the performance over the seven affordance labels considered.

From the results, we can see that between S-HMP and the SRF, S-HMP consistently outperforms SRF in all three evaluation metrics. The difference is most significant using the $F_\beta^w$ measure, which shows that the sparse codes obtained by S-HMP are able to distinguish the top ranked affordance class much better than SRF, which tends to produce weaker responses across multiple affordance categories. This is not surprising since, unlike HMP which learns *new* high-level features from the raw data, SRF only selects the most discriminative input features. Furthermore, the CNN performs better than both S-HMP and the SRF by a large margin on all metrics. In the sections that follow, we describe ablation experiments that demonstrate

| Affordance | CNN $(F_\beta^w, R_\beta^w, \overline{\tau}_k)$ | HMP $(F_\beta^w, R_\beta^w, \overline{\tau}_k)$ | SRF $(F_\beta^w, R_\beta^w, \overline{\tau}_k)$ |
|---|---|---|---|
| `grasp` | 0.330, 0.196, 0.713 | 0.227, 0.124, 0.583 | 0.200, 0.122, 0.165 |
| `cut` | 0.221, 0.090, 0.831 | 0.160, 0.065, 0.754 | 0.072, 0.030, 0.724 |
| `scoop` | 0.225, 0.176, 0.705 | 0.165, 0.083, 0.519 | 0.114, 0.106, 0.446 |
| `contain` | 0.437, 0.231, 0.688 | 0.437, 0.222, 0.627 | 0.322, 0.178, 0.316 |
| `pound` | 0.261, 0.096, 0.732 | 0.257, 0.079, 0.609 | 0.072, 0.023, 0.595 |
| `support` | 0.376, 0.075, 0.636 | 0.297, 0.049, 0.462 | 0.098, 0.022, 0.509 |
| `wrap-grasp` | 0.232, 0.128, 0.563 | 0.208, 0.109, 0.482 | 0.156, 0.099, 0.482 |
| Mean | **0.298**, **0.142**, **0.695** | 0.250, 0.105, 0.563 | 0.165, 0.083, 0.435 |

Table 3.4: Performance over the clutter subset.

the contribution of geometric features and how they help in real-world scenarios with clutter, occlusions and viewpoint changes.

### 3.6.5 Performance in clutter and occlusions

In order to test the performance of the approach in real-world situations containing clutter, occlusions and viewpoint changes, we tested our approach over the clutter subset of the RGB-D Part Affordance dataset. Table 3.4 compares the performance of the approaches using the $(F_\beta^w, R_\beta^w, \overline{\tau}_k)$ metrics.

We show in Figure 3.4 a series of three frames illustrating the responses of HMP and SRF for two specific affordances: `contain` and `wrap-grasp`. Despite changes in viewpoint, the approaches make reasonable predictions, such as correctly predicting the inner surfaces of bowls and cups as `contain`. HMP exhibits precisely localized predictions, and SRF demonstrates generalization, such as predicting `wrap-grasp`

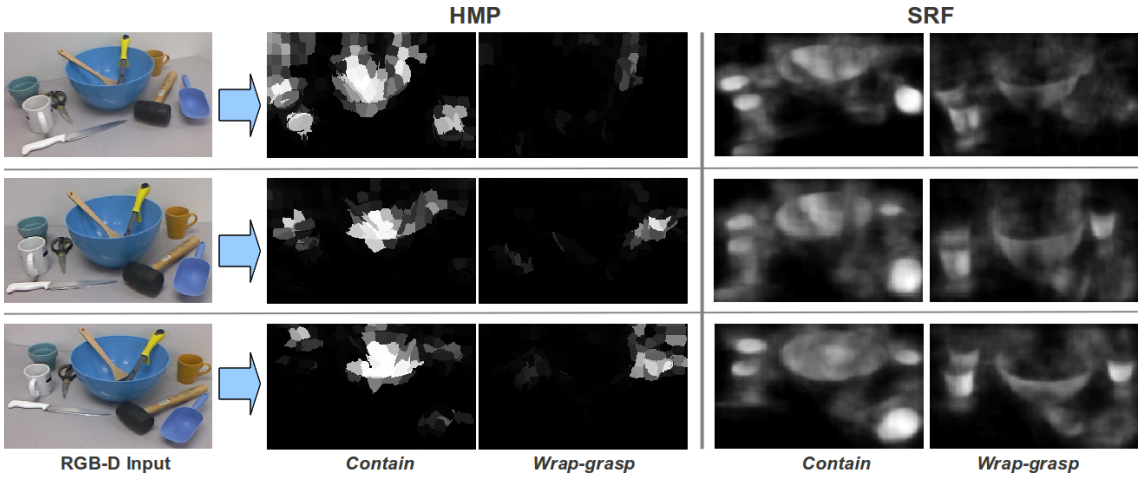



Figure 3.4: Results of affordance detection across three different input RGB-D frames (left) using HMP (middle) and SRF (right) over the cluttered sequence: two target affordances per method – `contain` (l) and `wrap-grasp` (r). Brighter means higher probability of the target affordance.

on the convex surface of the bowl. From Table 3.4, we note further that although both HMP and SRF's performance did drop under such challenging scenarios, the drop in HMP is less than SRF, which indicates that the learned features, unlike those obtained from SRF are far more robust to viewpoint changes and clutter than SRF. Still, the CNN is best performing method by far.

### 3.6.6 Ablation Experiments

We performed a series of feature ablations to demonstrate the contribution of each feature type in improving the results reported above. Table 3.5 shows the influence of additional features over the baseline smoothed and de-meaned depth features, denoted as `Depth`, with respect to the $F_\beta^w$ measure.

We see that the HMP baseline performs very well, and by learning multiple

| Feature Sets | CNN $F_\beta^w$ | HMP $F_\beta^w$ | SRF $F_\beta^w$ |
|---|---|---|---|
| `Depth` | 0.430 | 0.539 | 0.323 |
| `Depth+SNorm` | 0.659 (+0.228) | 0.547 (+0.008) | 0.444 (+0.121) |
| `Depth+SNorm+PCurv` | 0.663 (+0.232) | 0.562 (+0.023) | 0.449 (+0.126) |
| `Depth+SNorm+PCurv+SI+CV` | 0.672 (+0.241) | 0.557 (+0.018) | 0.460 (+0.137) |

Table 3.5: Ablation experiment results. $+x$ indicates the amount of change over `Depth`.

layers of features with increasing invariance and abstraction, HMP is able to extract discriminative information. Consequently, additional features provide better but diminishing returns on performance, consistent with the results in Section 2.7.2. One possible explanation of this effect is that increasing feature dimensionality could make SVM learning more difficult. Although the full set of features has a slightly lower $F_\beta^w$ measure, we note that it has the best performance on ranked measures and clutter. The SRF, on the other hand, benefits more from the addition of new features as they introduce more diversity into the random feature subsets used during training (Section 3.4). Using the full feature set the SRF achieves a large improvement over the ablated counterparts. Interestingly, we notice that although SI+CV are derived from PCurv, they improve the results further. This validates that the shape-index and curvedness measures capture discriminative information not provided directly by the other features. Finally, for the CNN we find that using only depth is significantly poorer performance than other feature subsets. Unlike for S-HMP and the SRF, the additional features are crucial, and each additional feature yields a noted performance improvement.

## 3.7 Conclusion

In this chapter, we have presented two methods for associating affordances with local shape and geometry information. These methods localize and identify multiple affordances of tool parts, providing functional information that can be used by a robot. The HMP method provides accurate results at a high computational cost, while SRF gives reasonable predictions in real-time. We have also demonstrated the importance of geometry for affordance identification, showing the importance of robust geometric features. We also validated our baseline on an existing dataset, achieving performance on par with a state-of-the-art deep learning approach, but at a fraction of the computation time. Finally, we extended the RGB-D Part Affordance Dataset with ranked affordance labels and cluttered 3 scenes.

# Chapter 4: Using Material Cues to Reason about Affordance

## 4.1 Introduction

The functionality of a tool not only depends on its geometric structure, but also on the underlying properties of the constituent materials. For example, if a robot needed a container to serve soup from a pot, we would expect it look for an object which has a concavity, and which is also made of a nonporous material. Therefore, it is important to utilize both geometry and material information to predict the affordances of tool parts. However, it's not immediately clear how material cues should be obtained and combined with geometry features to predict affordance. Building on the Convolutional Neural Network proposed in Chapter 3, we hypothesize that affordance predictions can be improved by integrating high level material cues, allowing the network to learn the relationship between geometry, affordance, and materials.

To this end, we propose to learn the relationships between materials and affordances using a two-stream deep convolutional neural network, with one stream for predicting affordance from geometric features, a second stream trained to recognize materials from RGB data, and a final stage to combine these high level cues. In this chapter, we first present the details of our approach in Section 4.3. In Section 4.4.1,

we introduce an Extended Part Affordance Dataset tailored towards material and affordance recognition. Lastly, we present experimental results which demonstrate the efficacy of the proposed method in Section 4.4.2.

## 4.2 Related Work

Texture classification has been long studied in the computer vision community, with early work on pattern recognition of grayscale images emerging half a century ago [46]. Since then, a wide variety of statistical, geometrical, model-based, and signal processing techniques have been proposed, with applications to such computer vision problems as face recognition, object recognition, action recognition, and background subtraction [47] [48]. Texture recognition laid the foundation for material recognition, but recognizing materials from images presents a much larger challenge, since it aims to infer the underlying physical properties of objects in images. Texture is often an important part of what defines a material's appearance, as is the case with materials like wood, leather, stone, and fabric which have unique textures. However, in many cases different materials may have similar textures, and in some cases objects made of the same material may have different textures. Consequently, common texture recognition approaches are not always well suited for material recognition [49–51].

Several prior works have addressed recognition of a specific material such as glass [52] and human skin [53]. As the field of material recognition has progressed, more comprehensive datasets have been developed to measure performance. The

CURET database [54] contains 61 material samples, each under 205 different lighting and viewing conditions. The KTH-TIPS2 database [55] was introduced with 11 material categories, 4 material sample instances per category, with each photographed under a variety of lighting conditions. The dataset was introduced to provide diverse material samples with greater intra-class variation, although the proposed SVM-based classifier was shown to achieve 98.5% accuracy on the database [55]. The Flickr Materials Database (FMD) introduced by [56] was one of the first material databases which leveraged internet images, consisting of 10 material categories, and 100 samples per category, hand picked from internet images on Flickr. Several approaches and hand engineered features have been proposed targeting the Flickr Materials Database including reflectance based edge features [57], features based on variances of oriented gradients [58], and pairwise local binary pattern (LBP) features [59].

Recently, and most relevant to this work, Bell *et al.* [60] introduced the Materials in Context (MINC) dataset with 3 million material samples mined from internet images on Flickr and images of staged interiors from professional photographers. MINC was developed from the OpenSurfaces dataset [61], which consists of 20,000 scenes and 105,000 segmentations from real world images. Unlike previous work on material recognition on the Flickr Materials Database where each image was associated with only a single label, MINC was designed for semantic segmentation, and the authors in [60] propose an approach based on deep convolutional neural networks with conditional random field refinement for the task. Lastly, a few recent works have shown that jointly predicting objects and materials can improve recognition

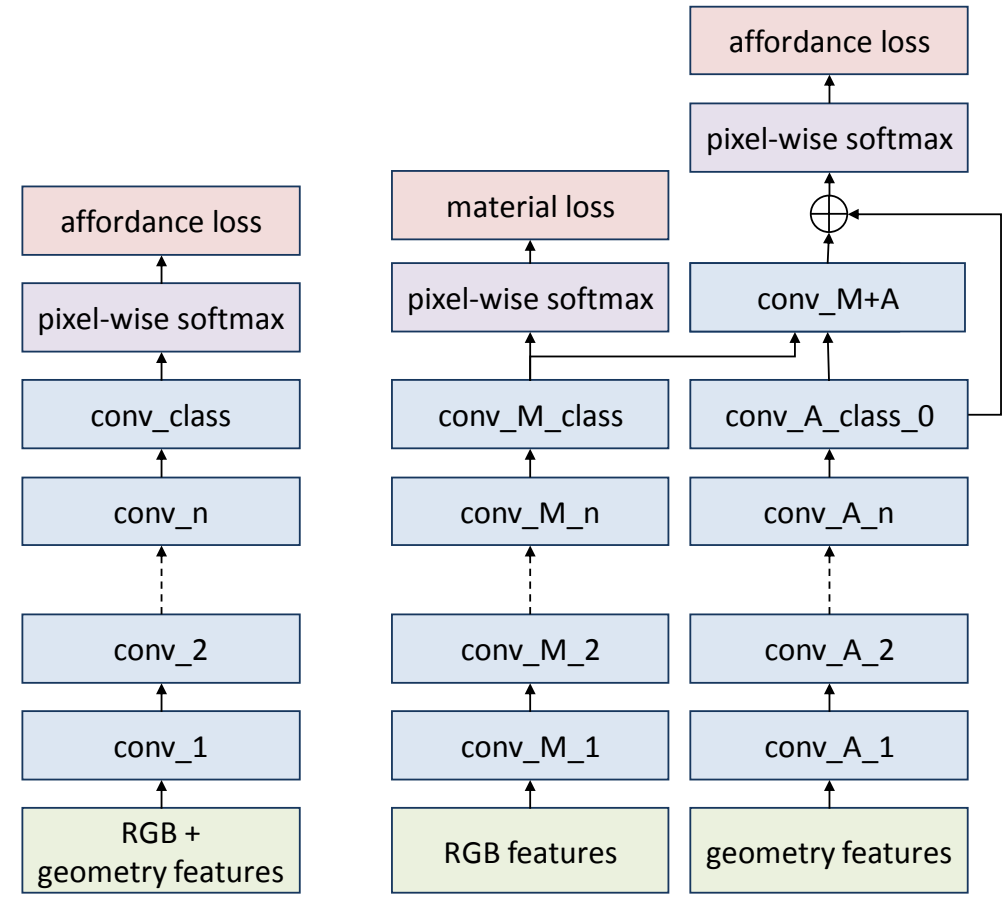
performance [58, 62].

## 4.3 Integrating Material Cues to Learn Affordance

The affordance of an object depends on both geometric and material properties, so naturally, we would expect any system for affordance prediction to utilize both of these cues. However, this raises questions regarding how geometry and material information should be combined, and their relative importance for the task. Intuitively, many objects in the world share similar materials, but only a few possess the necessary geometric features to satisfy a particular affordance. For example, a cutting tool could have a metal, ceramic, or even plastic blade, and a multitude of everyday objects are made of these materials, but only a few objects are sharp and thin enough to be used for cutting. So, the geometry of a part is generally the primary factor in determining its possible affordances, and this intuition is supported by experimental results from Chapters 2 and 3. Nevertheless, material properties still determine an object's affordances, as a sharp and thin object must be made a of a material hard enough to be used for cutting. Therefore, we propose to learn the relationships between materials and affordances using a two-stream deep convolutional neural network, with one stream for predicting affordance from geometric features, a second stream trained to recognize materials from RGB data, and a final stage to combine these high level cues.

The first stream is identical to the DCNN architecture proposed in Section 4.4. The network takes a tensor of geometric features as input, followed by 3 intermediate

convolution layers, and then a final convolution layer outputs an initial affordance prediction map $y_{A_0}$ with depth $k$, the number of affordance classes. Convolution layers are followed by ReLU non-linearities, and by overlapping $3 \times 3$ max-pooling with stride 2. As previously proposed, dilated convolution kernels are used to retain spatial resolution in the network output probability map. The second material stream follows the same structure as the first, except that it takes RGB data as input, and its output is a material class probability map $y_M$. The material stream is also followed by an auxiliary pixel-wise softmax layer and cross entropy loss. Both streams are finally combined through the element-wise addition of the initial affordance prediction map and the output of convolutions of the two stream's outputs, $y_A = y_{A_0} + F(y_{A_0}) + F(y_M)$. Intuitively, this should allow the network to learn the relationships between materials and affordances and use this information to to refine the initial affordance predictions $y_{A_0}$. The final affordance output $y_A$ is then followed by a pixel-wise softmax layer and cross entropy loss. The overall network architecture is illustrated in figure 4.1. The network can be trained end-to-end using the loss on the final affordance predictions and the auxiliary loss on the material prediction, or we can instead pretrain the material stream of the network first, on possibly a different dataset, before learning to predict affordance. This approach also has the advantage that gradient information from the material prediction will not be passed to the geometry stream of the network, and we can enforce that gradients are not passed to the material stream as well. This way we can include material information without affecting the performance of the material network, and other modalities could be added with their own objectives.




(a) Single Stream CNN (b) Two-Stream CNN

Figure 4.1: CNN architectures for affordance prediction using material cues. Single stream CNN (a) and the proposed two-stream CNN (b).

## 4.4 Experiments

We conduct experiments to evaluate the proposed approach and compare against the CNN proposed in Section as a baseline. In Section 4.4.1, we describe extensions made to the part affordance dataset to add new objects and material labels, and we present and discuss the results of our experiments in Section 4.4.2.

### 4.4.1 Extended Affordance and Material Dataset

The RGB-D Part Affordance Dataset, detailed in Section 2.6, consists of over 100 kitchen, workshop, and garden tools from 17 different categories. The tools in the dataset are made of a variety of materials, where even different instances from the same tool category are composed of different materials. For, example the dataset includes knives with plastic, metal, and wooden handles and even metal, ceramic, and plastic blades. However, one shortcoming of the dataset is that is does not include any non-tool objects. Recognizing when an object does not have any of the target affordances is a necessary ability in a robotics setting, and in many cases material information is an important cue to make such a distinction. For example, common office object made of paper or fabric might result in falsely detected affordances if material information is not properly used, so it is important that we train and test on such objects.

Therefore we extended the dataset by adding 123 non-tool objects from the UW RGB-D Object Dataset [32], which provides color and depth data of objects commonly found in home and office environments. Similar to the part affordance

dataset, the data is collected using a Kinect sensor, and objects are captured from multiple views on a rotating turntable. Approximately 250 images are collected for each object. A major difference between this data and the part affordance data is that the objects do not have labeled parts. Most of the objects are everyday non-tool objects such as fruits, food boxes, office supplies, and cleaning supplies. In choosing objects to expand our dataset, we selected those that clearly could not be used for any of our 7 affordances.

Data in both the RGB-D Part Affordance Dataset and the UW RGB-D Object Dataset [32] is collected using the Kinect sensor. Due to the limitations of the Kinect sensor, images are taken from approximately 1 meter away, and thus for some objects it is difficult to capture detailed material information. In addition, the Kinect sensor is primarily designed for human body gesture recognition in low light settings for video game applications, so the sensor's color images are not high quality.

So, in addition to adding new non-tool object data, we also collected new data from another 58 objects using a Intel Realsense Softkinetic RGB-D Camera. The Softkinetic sensor is designed for hand gesture recognition and person computer interaction applications at close range. In contrast to the Kinect sensor which has a minimum range of approximately 0.8 meters, the Softkinetic sensor has a minimum range closer to 0.3 meters. Furthermore, while the Kinect sensor estimates depth from projected structured infrared light patterns, the Softkinetic sensor uses a time-of-flight approach which can capture fine details. Following the same procedure described in 2.6, we recorded 58 new tool and non-tool objects on a revolving turntable to collect images covering a full 360° range of views, and we labeled tools

and their parts by ranking their affordances. Finally, for all of the data collected, we also labeled the objects' materials using the same labeling tool used to label affordances. The material labels in the dataset include, *ceramic*, *fabric*, *food*, *metal*, *paper*, *plastic*, *rubber*, *sponge*, and *wood*.

### 4.4.2 Results

We report results that demonstrate the performance using the metrics $(F_\beta^w, R_\beta^w, \overline{\tau}_k)$ for the baseline CNN and the CNN+material information. We use a 5-fold training and testing methodology to split the Extended RGB-D Part Affordance dataset described in Section 4.4.1 into 5 folds. The data is split by object, resulting in approximately 12,000 images for training and 3000 images for testing per fold. For each of the metrics, we report the average over all 5 folds. We used the same geometric features for each network, and for simplicity we use the depth and surface normals since this combination performed well in our experiments in Chapter 3. Table 4.1 summarizes the two methods' performance over the seven affordance labels considered.

From the results, we can see that the material network outperforms the baseline in all evaluation metrics. The difference is most significant using the $F_\beta^w$ measure, which shows that the integration of material information helps distinguish the top ranked affordance class much better than the baseline. We show example results for both methods in Figure 4.2. We can see that the material network makes better predictions in general, but specifically in cases where the tool parts are made of

| Affordance | CNN ($F^w_\beta, R^w_\beta, \overline{\tau}_k$) | CNN+Material ($F^w_\beta, R^w_\beta, \overline{\tau}_k$) |
|---|---|---|
| `grasp` | 0.211, 0.088, 0.950 | 0.282, 0.117, 0.950 |
| `cut` | 0.111, 0.013, 0.989 | 0.335, 0.039, 0.986 |
| `scoop` | 0.167, 0.029, 0.947 | 0.271, 0.047, 0.959 |
| `contain` | 0.711, 0.153, 0.963 | 0.628, 0.135, 0.970 |
| `pound` | 0.123, 0.008, 0.957 | 0.186, 0.012, 0.965 |
| `support` | 0.134, 0.009, 0.969 | 0.225, 0.016, 0.967 |
| `wrap-grasp` | 0.776, 0.103, 0.987 | 0.723, 0.096, 0.990 |
| `no affordance` | 0.991, 0.290, 0.991 | 0.992, 0.290, 0.991 |
| Mean | 0.403, 0.087, 0.969 | **0.4551**, **0.094**, **0.972** |

Table 4.1: Results on the Extended RGB-D Part Affordance Dataset for the baseline CNN and the CNN with material information.

metal or dark materials, where depth data is unreliable. For example, we can see that in the case of the knife, the network without material cues does not predict that the metal blade can be used for cutting, while the network using material correctly identifies the affordance in this case. Similarly, in the cases with the spatula and mallet, the objects' dark colors can result in noisy depth data, so we see marked improvements when material information is used.

## 4.5 Conclusion

The functionality of a tool depends on both its geometric and material properties. In this chapter, we have proposed to learn the relationships between materials and affordances using a two-stream deep convolutional neural network, with one stream for predicting affordance from geometric features, a second stream trained to recognize materials from RGB data, and a final stage to combine these high

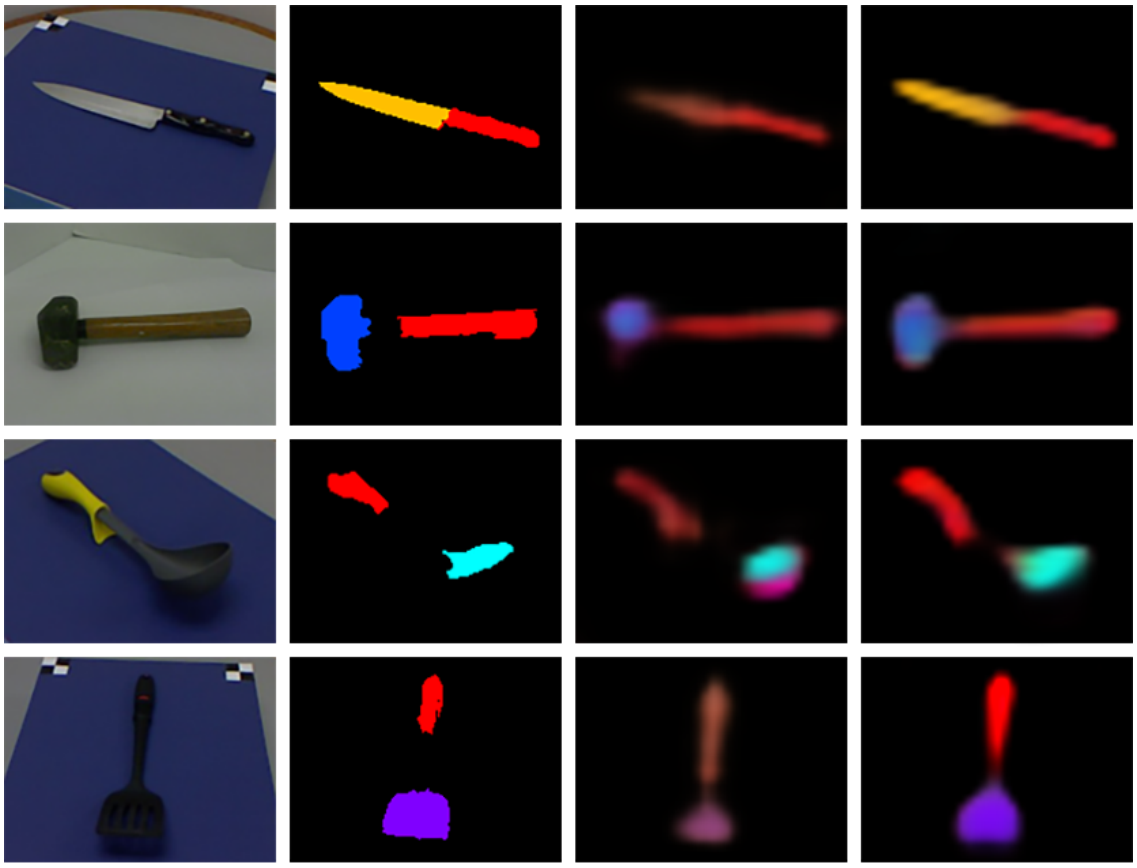



Figure 4.2: Examples results for the methods CNN and CNN+material. Original color images (a), ground truth affordance labels (b), predictions from the CNN (c), and predictions from the CNN with material cues (d).

level cues. In addition, we developed an Extended Part Affordance Dataset tailored towards material and affordance recognition, with 181 addition objects from multiple types of depth sensors. Through experiments we showed that integrating high level material cues allowing the proposed approach to learn the relationship between geometry, affordance, and materials, thereby improving affordance predictions compared to other approaches. Ultimately, material recognition cannot always be solved by vision alone, and other sensing modalities may need to be employed for robust performance in robotics applications. For example, a robot looking for a metal fork might be fooled by a plastic fork with a chrome paint finish, since it would be nearly indistinguishable visually. The proposed approach is also suitable for mid-level cues other than materials and presents interesting directions for future work.

# Chapter 5: Conclusion

In this thesis, we propose the novel problem of predicting part affordances of tools. This is in contrast to previous approaches in image classification and robotic applications that predominantly focuses on instance or object level classification. This work opens up possibilities for robots to identity novel objects it has not seen before and understand their functionality. Since tools can have more than one affordance, we propose ranked affordance labels, where an object is assigned a multiple affordances, ordered by its significance. We propose multiple datasets for the evaluation of this task for tools in a cluttered and occluded environments with material information and per-pixel ground truth labels and rankings. We also propose several evaluation metrics for this task. We hope this work opens up exciting new research directions for reasoning about novel objects and manipulating them by their affordance.

# Appendix A: Im2Calories: towards an automated mobile vision food diary

While this thesis presents the problem of predicting part affordances, the approaches proposed therein are also applicable to other similar research problems. During the summer of 2015, I interned at Google Research, where I worked with Kevin Murphy's team on a project to estimate the number of calories in an image of food. I used many of the semantic segmentation and deep learning techniques and lessons learned from my affordance work in order to solve this problem. Estimating the nutritional information of food from an image also differs from many more common vision problems, since it's necessary to recoginze the parts of various foods and ingredients in order to produce a precise segmentation. In this appendix, we include the paper describing this work.

# Im2Calories: towards an automated mobile vision food diary

Austin Myers, Nick Johnston, Vivek Rathod, Anoop Korattikara, Alex Gorban
Nathan Silberman, Sergio Guadarrama, George Papandreou, Jonathan Huang, Kevin Murphy
amyers@umd.edu, (nickj, rathodv, kbanoop, gorban)@google.com
(nsilberman, sguada, gpapan, jonathanhuang, kpmurphy)@google.com

We present a system which can recognize the contents of your meal from a single image, and then predict its nutritional contents, such as calories. The simplest version assumes that the user is eating at a restaurant for which we know the menu. In this case, we can collect images offline to train a multi-label classifier. At run time, we apply the classifier (running on your phone) to predict which foods are present in your meal, and we lookup the corresponding nutritional facts. We apply this method to a new dataset of images from 23 different restaurants, using a CNN-based classifier, significantly outperforming previous work. The more challenging setting works outside of restaurants. In this case, we need to estimate the size of the foods, as well as their labels. This requires solving segmentation and depth / volume estimation from a single image. We present CNN-based approaches to these problems, with promising preliminary results.

## 1. Introduction

Many people are interested in tracking what they eat to help them achieve weight loss goals or manage their diabetes or food allergies. However, most current mobile apps (MyFitnessPal, LoseIt, etc) require manual data entry, which is tedious and time consuming. Consequently, most users do not use such apps for very long [9]. Furthermore, amateur self-reports of calorie intake typically have an error rate that exceeds 400 calories per day [31, 5].

Rather than rely purely on data entry, several approaches make use of a mobile camera to aid in this task. Cordeiro *et al*. [8] records a photo of the meal but does not perform any visual analysis of the image. Several previous approaches [23] [29] rely on an expert nutritionists to analyse the image offline (at the end of each day). Other approaches [26] [25] use crowd sourcing to interpret the image, in lieu of an expert. However, crowd sourcing is both costly and slow, which hinders widespread adoption.

Several existing works [39, 21, 37] do use computer vision algorithms to reason about meals but only work in laboratory conditions where the food items are well separated and the number of categories is small. Furthermore, most of these methods use traditional, hand-crafted visual features, and only use machine learning at the classification stage.

The holy grail is an automatic method for estimating the nutritional contents of a meal from one or more images. Unfortunately, even a perfect visual interpretation of the scene cannot perfectly predict what is inside many foods, e.g., a burrito. Consequently, an ideal system is one which correctly infers what is knowable and prompts the user for feedback on inherantly ambiguous components of a meal. Our goal is to minimize user effort for completing food diaries by offering smart "auto-complete" functionality, rather than complete automation.

In this paper, we take some initial steps towards such a system. Our approach utilizes several deep learning algorithms, tailored to run on a conventional mobile phone, trained to recognize food items and predict the nutrional contents meals from images taken "in the wild". We refer to this task as the "Im2Calories" problem, by analogy to the recent line of work on the "Im2Text" problem. It should be stressed, however, that we are interested in estimating various other properties of a meal (such as fat and carbohydrates) and not just calories.

We start by building on [1], who developed a system that can predict the calorie content of a meal from a single image. Their key insight (also made in [2]) was that the problem becomes much simpler if we restrict the setting to one where the user is eating in a restaurant whose menu is known. In this case, the problem reduces to one of detecting which items, out of the $K$ possible items on the menu, the user has chosen. Each item typically has a standard serving size[1], and we typically know its nutritional contents.[2], whereas getting nutritional information for arbitrary cooked foods is much harder, as discussed in Section 7.

In Section 3, we show that by simply replacing the hand-crafted features used in [1] with a convolutional neural network (CNN), we can significantly improve performance,

[1] There may be variants, but these are typically few in number (e.g., small, medium or large fries). Hence we an treat these as different items.

[2] The US Food and Drug Administration has passed a law requiring all major chain restaurants to post the nutritional contents of their meals, starting in December 2016. See [15] for details.

both at labeling the foods and estimating total calories. We then extend their method from 3 restaurants to 23 restaurants, increasing the coverage from 41 food items to 2517. We show that the same basic multilabel classification system continues to work.

Unfortunately, it is hard to get enough images for all the menu items for all the restaurants in the world. And even if we could, this would not help us when the user is not eating at a restaurant. Therefore, in Section 4, we develop a set of 201 generic, restaurant-independent food tags. We then extend the existing public Food101 dataset [3] with these tags using crowdsourcing. We call the resulting dataset Food201-multilabel and plan to release it publicly.[3] We show that the same kind of CNN-based multi-label classifier also works fairly well on this new (larger and more challenging) dataset, although we found it necessary to perform some clustering of visually indistinguishable labels in order to get acceptable performance.

Of course, detecting the presence of a food item in an image is not sufficient, since most items can be "parameterized" in terms of size, toppings, etc. Note that this is true even in the restaurant setting. We could of course ask the user about these variants, but we would like to do as much as possible automatically.

To be able to perform such fine grained classification, we need to be able to localize the objects within the image and extract detailed features. We argue that a segmentation-based approach is more appropriate than the traditional bounding-box approach, since food items can have highly variable shape. In Section 5, we present the new Food201-segmented dataset, and our approach to semantic image segmentation. We show that leveraging the multilabel classifier from the earlier stage can help with segmentation, since it provides a form of "context".

Once we have segmented the foods, we can try to estimate their volume. To do this, we need to know the surface height of the foods above the plate. In Section 6, we present some preliminary results on estimating the depth of each pixel from a single RGB image, using a CNN. We then show promising results on estimating the volumes of foods.

In summary, this paper makes 3 main contributions. First, we develop a system that can recognize the contents of a restaurant meal much more accurately than the previous state of the art, and at a much larger scale. Second, we introduce a new dataset, Food201, and show how it can be used to train and test image tagging and segmentation systems. Third, we show some promising preliminary results on the challenging problem of mapping image to calories from images taken in the wild, in a non-restaurant setting. Our overall system is illustrated in Figure 1.

[3] See https://storage.googleapis.com/food201/food201.zip.

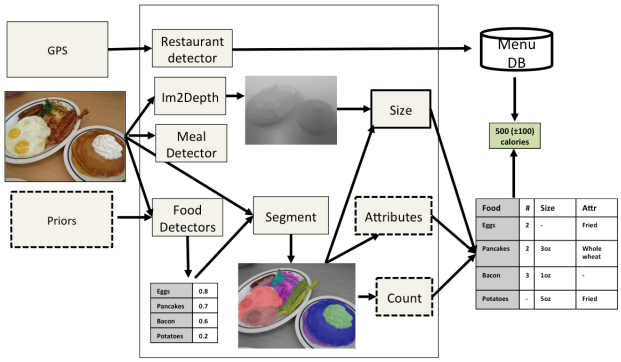



Figure 1. Illustration of the overall system. Dotted boxes denote components that have not yet been implemented. The input is one or more images, and optionally a GPS signal and user priors (e.g., concerning food preferences). The output is a food diary, and an estimated total calorie count (in cases where a suitable nutritional database is available, such as from a restaurant's menu).

| Name | #Classes | #Train | #Test | Comments |
|---|---|---|---|---|
| Food101 | 101 | 75,750 | 25,250 | [3] |
| Food101-Background | 2 | 151,500 | 50,500 | Food vs non-food |
| Food201-MultiLabel | 201 | 35,242 | 15,132 | Multi label |
| Food201-Segmented | 201 | 10,100 | 2,525 | Label per pixel |
| Restaurant | 2517 | 75k | 25k | Single label |
| Gfood-3d | - | 150k | 2471 | Depth per pixel |
| Nfood-3d | - | - | 1050 | Depth per pixel |

Table 1. Summary of the datasets used in this paper. The Restaurant dataset contains web images for 23 restaurants. Gfood-3d is from 50 Google meals. Nfood-3d is from 11 meals made with Nasco food replicas.

## 2. Meal detection

The first step in our pipeline is to determine if the image is an image of a meal at a "reasonable" distance from the camera. We can phrase this as a simple binary classification problem. To tackle this problem, we need to have a suitable dataset. We start by taking the Food101 dataset developed in [3], which contains 101 classes of food, 1000 images each (750 train, 250 test). Food101 is the largest publicly available dataset of food images we are aware of (see Table 1 for a comparison with some other public datasets).

The Food101 dataset is designed for multi-class classification. To make a dataset suitable for binary classification, we combined all the food classes into one generic "food" class, and then randomly extracted an equal number of images from the ImageNet challenge dataset [30] to create the "non-food" class.[4]

Each image is rescaled (if necessary) so that its maximum height or width is 512 pixels, but preserving the original aspect ratio. We call this new dataset the **Food101-**

[4] ImageNet actually contains 51 food classes, so we removed these images from the negative set.

**Background** dataset.

To train a classifier for this problem, we took the GoogLeNet CNN model from [34], which had been pre-trained on ImageNet, removed the final 1000-way softmax, and replaced it with a single logistic node. Then we fine tune the whole model on the Food101-Background dataset; this takes about 12 hours using a single Titan X GPU with 12 GB of memory. The final test set accuracy is 99.02%.

## 3. Restaurant-specific im2calories

Once we have determined that the image contains a meal, we try to analyze its contents. The first step is to determine which restaurant the user is in. In this paper, we use Google's Places API [27] for this. We then retrieve the menu of the nearest restaurant from the web,[5] parse the menu into a list of $K$ food items, and retrieve images for each of these, either by performing web search (as in [2]) or by asking users to collect images (as in [1]).[6]

Once we have the dataset, we can train a classifier to map from image to label. Since the user may have multiple food items on their plate, it is better to use a multi-label classifier rather than using a multi-class classifier, which assumes the labels are mutually exclusive. Next we can estimate the set of foods that are present by picking a suitable threshold $\phi$, and then computing $\mathcal{S} = \{k : p(y_k = 1|x) > \phi\}$, where $p(y_k = 1|x)$ is the probability that food $k$ is present in image $x$. Finally, we can lookup each of these food items in our (restaurant specific) database, and then estimate the total calories as follows: $\hat{C} = \sum_{k \in \mathcal{S}} C_k$, where $C_k$ is the calorie content of menu item $k$. Alternatively, to avoid having to specify the threshold $\phi$, we can compute $\bar{C} = \sum_{k=1}^{K} p(y_k = 1|x) C_k$. We compare these methods below.

### 3.1. Experiments on MenuMatch dataset

To evaluate this approach, we used the dataset from [1], known as "MenuMatch". This consists of 646 images, tagged with 41 food items, taken from 3 restaurants. Unlike other datasets, each image in MenuMatch has a corresponding ground truth calorie estimate, computed by an expert nutritionist. In addition, each restaurant's menu has corresponding ground truth calorie content per item.

[1] used various hand-crafted features together with a linear SVM, trained in a one-vs-all fashion. Their best performing system achieved a mean average precision (mAP) of 51.2% on the test set. By contrast, we get much better results using a deep learning approach, as we explain below.

| Method | Mean error | Mean absolute error |
|---|---|---|
| Baseline | $-37.3 \pm 3.9$ | $239.9 \pm 1.4$ |
| Meal Snap | $-268.5 \pm 13.3$ | $330.9 \pm 11.0$ |
| Menu Match | $-21.0 \pm 11.6$ | $232.0 \pm 7.2$ |
| $\hat{C}$ | $-31.90 \pm 28.10$ | $163.43 \pm 16.32$ |
| $\bar{C}$ | $-25.35 \pm 26.37$ | $152.95 \pm 15.61$ |

Table 2. Errors in calorie estimation on the MenuMatch dataset. $\hat{C}$ and $\bar{C}$ are our methods. Numbers after the $\pm$ sign are standard errors estimated by 5-fold cross-validation. See text for details.

We took the GoogLeNet CNN model from [34], which had been pre-trained on ImageNet, removed the final 1000-way softmax, replaced it with a 101-way softmax, and fine-tuned the model on the Food101 dataset [3]. The resulting model has a classification accuracy on the Food101 test set of 79%, which is significantly better than the 50.76% reported by [3] (they used hand crafted features plus an SVM classifer using a spatial pyramid matching kernel).

Next, we took the model which was trained on Food101, removed the 101-way softmax, replaced it with 41 logistic nodes, and fine-tuned on the MenuMatch training set. (Pre-training on Food101 was necessary since the MenuMatch dataset is so small.) The resulting mAP on the test set is 81.4%, which is significantly higher than the best result of 51.2% reported in [1].

Finally, we wanted to assess the accuracy of calorie prediction. We compared 5 methods: the MenuMatch system of [1], the MealSnap app [25] that uses crowdsourcing, our method using $\hat{C}$, our method using $\bar{C}$, and finally, a baseline method, which simply computes the empirical mean of the calorie content of all meals from a specific restaurant. The results are shown in Table 2. We see that our system has considerably lower error than MenuMatch and the crowdsourced MealSnap app. (The unreliability of MealSnap was also noted in [26].) In fact, we see that MenuMatch barely beats the baseline approach of predicting the prior mean.

### 3.2. Scaling up to more restaurants

The MenuMatch results are based on 646 images of 41 food items from 3 restaurants. In this Section, we discuss our attempts to scale up these experiments.

First, we downloaded the menus for the top 25 restaurants in the USA, as ranked by sales.[7] We decided to drop "Pizza Hut" and "Chipotle", since gathering images for their menu items was tricky.[8] From the remaining 23

[5] This data is available for many US chain restaurants in semi-structured form from https://www.nutritionix.com.

[6] In practice, it is suprisingly complicated to parse the menus and retrieve relevant images. For example, the restaurant "16 handles" contains the folowing items: NSA Blueberry Tease, NSA Chocolate Eruption, NSA Strawberry Fields, and 65 other similar entries. You need to know that these are from the frozen yogurt section of the menu, and that NSA stands for "no sugar added", in order to make any sense of this data.

[7] Source: http://nrn.com/us-top-100/top-100-chains-us-sales.

[8] For example, "Pizza Hut" has menu items such as "chicken", "pepperoni", etc. But these are toppings for a pizza, not individual food items. Similarly, "Chipotle" lists "chicken", "beans", etc. But these are fillings for a burrito, not individual food items.

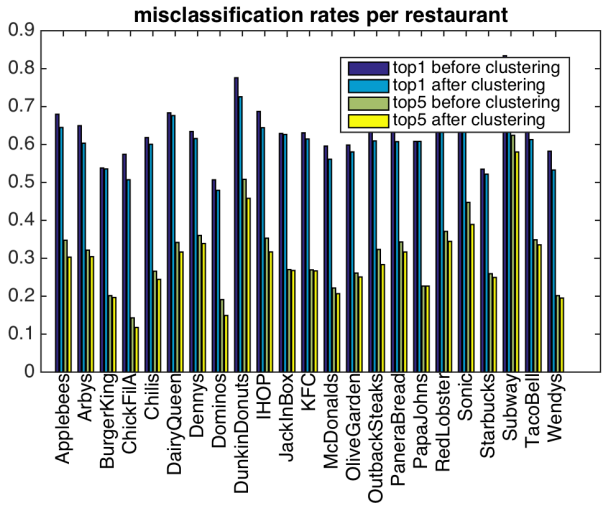



Figure 2. Top 1 and top 5 error rates on the test set for 23 different restaurants, before and after clustering the most confusable labels.

restaurants, we collected 4857 menu items. We manually removed drinks and other miscellaneous items, and then scraped 1.2 million images by issuing image search queries of the form "<restaurant name> <item name> (yelp | flickr | instagram | pinterest | foodspotting)". (These site restricts were chosen to increase the chances that we collected user-generated photos, rather than "official" photos from the restaurant itself, which tend to be too easy.) Of the scraped images, 270k were sent to Amazon Mechanical Turk for verification, producing a final set of 2517 menu items and 99k images. We call this the **Restaurant** dataset.

We then took our GoogleLeNet model, which had been trained on ImagetNet and then Food101, and replaced the 101-softmax with 2517 logistic nodes. We trained the final layer of this model on 75% of the data, and tested on the rest. We then computed the top 1 and top 5 error rates, averaged over the food items, for each of the 23 restaurants. The results are shown in Figure 2.

The top 1 error rate is quite high. This is because many food items are extremely similar, and it is hard, even for a human expert, to tell them apart. For example, McDonalds has the following items on its menu: Quarter Pounder Deluxe, Quarter Pounder Bacon Cheese, Quarter Pounder with Cheese, etc. (See Figure 3 for an illustration of these food items.)

To combat this issue, we computed the class confusion matrix on the training set for each restaurant, and then performed a very simple clustering of similar items. In particular, we computed the $K$ nearest neighbor graph, in which we connected each label to the $K$ other labels it is most often mapped to (which can include itself). We then computed the connected components to form a set of clusters.

Figure 3. The first two images are put into the same visual cluster, the third is kept distinct. Image sources: http://www.mcdonaldsmenu.mobi/beefburgersmenu/deluxequarterpounder/. http://www.mcdonaldsmenu.mobi/beefburgersmenu/baconcheesequarterpounder/. http://www.mcdonalds.com/us/en/food/product_nutrition.burgerssandwiches.7.quarter-pounder-with-cheese.html

We found qualitatively that using $K = 1$ gave a good tradeoff between merging the most confusable classes and overclustering. With $K = 1$, the number of clusters was about 0.9 times the original number of items; most clusters were singletons, but some had 2 or 3 items in them. Figure 3 gives an example of two clusters we created from the McDonald's menu; the first cluster contains two very visually similar items (Quarter Pounder Deluxe and Quarter Pounder Bacon Cheese), and the second cluster contains a visually distinctive item (Quarter Pounder with Cheese).

Finally, we evaluated performance of the classifier on the clustered test labels, by defining the probability of a cluster as the max probability of any label in the cluster. Figure 2 shows that, not surprisingly, the error rates decrease. In the future, we hope to try using a hierarchical classifier, which can tradeoff specificity with error rate c.f., [11].

## 4. Generic food detection

The results from Section 3 required images for each menu item from each restaurant. It is difficult to acquire such data, as previously remarked. To create a more generic dataset, we took half of the Food101 dataset (50k images), and asked raters on Mechanical Turk to name all the food items they could see in each image. We included the original class label as one of the choices, and manually created a list of commonly co-occuring foods as additional choices in the drop-down menu (e.g., eggs often co-occur with bacon). We also allowed raters to enter new terms in a text box. We used 2 raters per image. After manually merging synonymous terms, and removing terms that occurred less than 100 times, we ended up with a vocabulary of 201 labels. On average, each image had 1.9 labels (with a median of 1 and a max of 8).

We split the resulting dataset into 35,242 training images and 15,132 test images, in a way which is consistent with the Food101 train/ test split. We call this the **Food201-MultiLabel** dataset. See Table 1 for a summary of how this dataset compares to other public food datasets.

Next, we developed a multi-label classifier for this

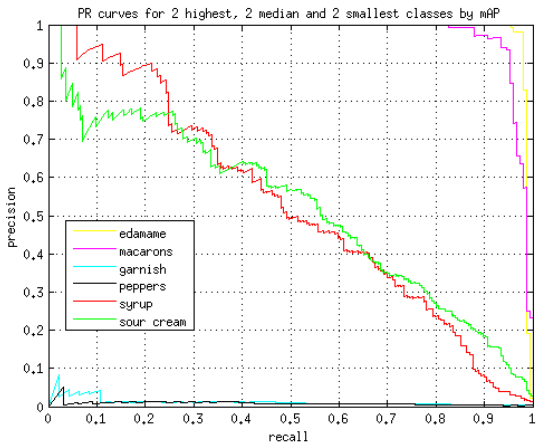



Figure 4. Precision-recall curves for 6 classes, ranging from best to worst, on the Food201-MultLabel dataset.

dataset, using the same method as in Section 3 (namely taking the GoogLeNet model, replacing the 101-way softmax with 201 logistic nodes). This takes about half a day to train on a single GPU. We then compute the average precision (area under the precision-recall curve) for each class, and average this over the classes, to compute the mean average precision (mAP).

The mAP is 0.8 for classes in Food 101, 0.2 for classes outside Food 101, and 0.5 overall. Not surprisingly, we do better on the original Food101 classes, since the new classes often correspond to side dishes or smaller food items, and are much less frequent in the training data. The top 3 classes were: edamame (0.987), macarons (0.976), hot and sour soup (0.956). The bottom 3 classes were: cream (0.015), garnish (0.014), peppers (0.010). Figure 4 shows the precision-recall curves for some of these classes.

## 5. Semantic image segmentation

In addition to predicting the presence of certain foods, it is useful to localize them in the image. Since most foods are amorphous, it is better to segment out the region corresponding to each food, rather than putting a bounding box around them. Such a segmented image will enable further analysis, such as counting and size estimation (see below), which is essential for nutrition estimation. We can also allow the user to interactively edit the estimated segmentation mask, in order to improve accuracy of the system (although we leave this to future work).

To train and evaluate semantic image segmentation systems, we took a subset of the Food201-MultiLabel dataset (12k images), and asked raters on a crowd computing platform to manually segment each of the food items that have been tagged (in an earlier stage) in that each image. We used 1 rater per image, and an internal tool that leverages grabcut to make this labeling process reasonably fast. Raters had the option of skipping foods that were too hard to segment. Thus some foods are not segmented, resulting in false negatives. We call this the **Food201-segmented** dataset.

Note that we are segmenting each class, as in the PASCAL VOC semantic segmentation challenge [13]. Arguably, instance-level segmentation (see e.g., [17]) would be more useful, since it distinguishes different instances of the same class, which would enable us to count instances.[9]. However, this is quite difficult. For example, consider distinguishing pancake 1 from pancake 2 in the food image in Figure 1: this is quite challenging, since the top pancake almost completely covers the bottom one. Furthemore, segmenting the eggs into 2 instances is even harder, since the boundary is not well defined. We leave instance-level segmentation to future work.

To tackle the segmentation problem, we use the "DeepLab" system from [6].[10] This model uses a CNN to provide the unary potentials of a CRF, and a fully connected graph to perform edge-sensitive label smoothing (as in bilateral filtering).

The model is initialized on ImageNet, and then fine-tuned on Food201-segmented, which takes about 1 day on a single GPU. For the 3 CRF parameters, which control the strength of the edge potentials, we use the parameter values from [6]; these were chosen by grid search, to minimize validation error on held-out training data.

The label set in the Food201-Segmented dataset is much larger than in the VOC challenge (201 labels instead of just 20). Consequently, the baseline model has a tendency to generate a lot of false positives. To improve performance, we take the probability vector $p(y_k = 1|x)$ computed by the multi-label classifier from Section 4, find the top 5 entries, and then create a binary mask vector, which is all 0s except for the top 5 labels, plus the background. We then multiply the per-pixel label distribution from the segmentation CNN by this sparse vector, before running the CRF smoothing. This provides a form of global image context and improves the results considerably (see below).

We show some sample results in Figure 5. Consider the last row, for example. We see that the context from the multilabel model helps by eliminating false positives (e.g., caprese salad and caesar salad). We also see that the ground truth is sometimes incomplete (e.g., the burrito is not actu-

[9] It is more natural for the user if the system records in their food diary that they ate 3 slices of pizza rather than, say, 120 ounces of pizza. It is also much easier for the user to interactively fix errors related to discrete counts than continuous volumes. Of course, this assumes we know what the size of each slice is. We leave further investigation to future work.

[10] At the time this paper was submitted, DeepLab was the best performing method on the PASCAL VOC challenge (see http://host.robots.ox.ac.uk:8080/leaderboard/displaylb.php?challengeid=11&compid=6). At the time of writing the camera ready version, the best performing method is an extension of DeepLab that was additionally trained on MS-COCO data.

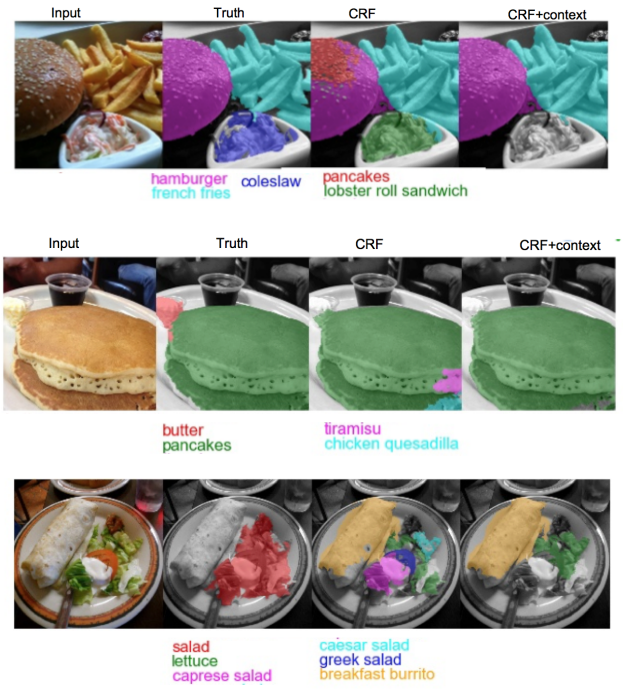



Figure 5. Examples of semantic image segmentation on some images from the Food201-Segmented test set. First column: original image. Second column: ground truth. Third column: predictions using CNN/CRF. Fouth column: predictions using CNN/CRF with multilabel context. Best viewed in color.

| CRF? | Context? | Acc | Recall | IoU |
|---|---|---|---|---|
| 0 | 0 | 0.71 | 0.30 | 0.19 |
| 1 | 0 | 0.74 | 0.32 | 0.22 |
| 0 | 1 | 0.74 | 0.32 | 0.23 |
| 1 | 1 | 0.76 | 0.33 | 0.25 |

Table 3. Performance on the Food101-Segmented test set.

ally labeled by the human). Finally, we see that the label space is somewhat arbitrary: the ground truth uses the term "salad", whereas the model predicts "lettuce". Currently we ignore any semantic similarities between the labels.

The performance of our system with and without the CRF, and with and without multilabel context, is shown in Table 3. The performance in terms of IoU is much lower than on the PASCAL VOC 2012 segmentation challenge. We believe this is due to several factors: (1) we have 201 foreground labels to predict, whereas VOC just has 20; (2) our labeled dataset suffers from incompleteness, which can result in correct predictions being erroneously being counted as false positives, as well as "polluting" the background class during training; (3) our task is arguably intrinsically harder, since the categories we wish to segment are more deformable and varied in their appearance than most of the VOC categories.

## 6. Volume estimation

Having segmented the foods, the next step is to estimate their physical size (3d volume).

We first predict the distance of every pixel from the camera, using the same CNN architecture as in [12] applied to a single RGB image. We trained our model on the NYU v2 RGBD dataset of indoor scenes, and then fine tuned it on a new 3d food dataset we collected which we call the **GFood3d** dataset (G for Google). This consists of short RGBD videos of 50 different meals from various Google cafes collected using the Intel RealSense F200 depth sensor.[11] In total, the dataset has about 150k frames. (Note that each RGB image has a corresponding depth image, but otherwise this is unlabeled data.)

On a test set of 2,471 images (recorded in a different set of Google cafes), we get an average relative error of 0.18 meters, which is slightly better to the performance reported in [12] on the NYU dataset. Figure 6 shows a qualitative example. We see that the CNN is able to predict the depth map fairly well, albeit at a low spatial resolution of 75 x 55. (For comparison with the F200 data, we upsample the predicted depth map).

The next step is to convert the depthmap into a voxel representation. To do this, we first detect the table surface using RANSAC. Next, we project each pixel into 3d space, exploiting the known intrinsic parameters of the F200 sensor. Finally, we tile the table surface with a 2d grid (using a cell size of 2mm x 2mm), and compute the average height of all the points in each cell, thus creating a "tower" of that height. For an example of the result of this process, see Figure 7.

Given a voxel representation of the food, and a segmentation mask, we can estimate the volume of each food item. To measure the accuracy of this approach, we purchased the "MyPlate" kit from Nasco.[12] This contains rubber replicas of 42 food items in standard sizes, and is used for training nutritionists. We verified the actual size of these items using the water displacement method (we found that the measured size was somewhat smaller than the quoted sizes). We then created 11 "meals" from these food replicas, and recorded images of them using the F200. Specifically, we put the meals on a turntable, and automatically took 100 images as the plate went through a full rotation. The resulting dataset has 1050 depth images. We call this the **NFood-3d** dataset (N for Nasco).

To measure performance of our system, we compute the absolute error in the volume estimate. We can estimate the volume using the true depth map (from F200) or the pre-

[11] The F200 has a spatial resolution of 1920 x 1080 pixels, and a sensing range of 0.2–1.2m. The Kinect2 camera has the same resolution, but a sensing range of 0.8–3.5m. Thus the F200 is a better choice for close up images.

[12] Source: http://www.enasco.com/product/WA29169HR.

Figure 6. (a) An image from the GFood-3d test set. (b) Depth recorded from RealSense RGBD sensor. (c) Depth predicted by CNN.

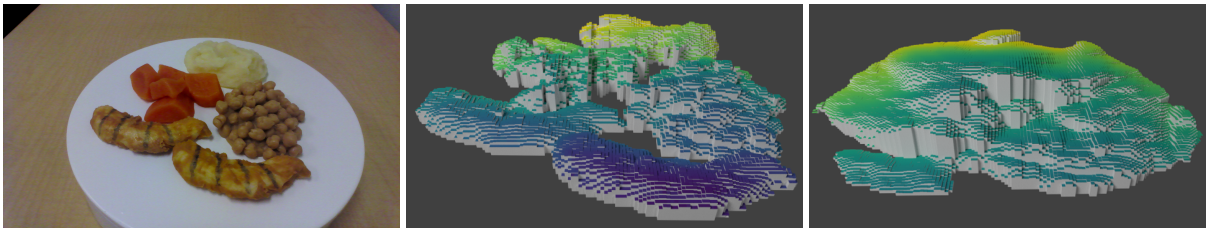


Figure 7. (a) An image from the NFood-3d dataset. (b) Voxel grid derived from RealSense depthmap. (c) Voxel grid estimated from CNN predicted depthmap. Size of grid cell is 2mm x 2mm.

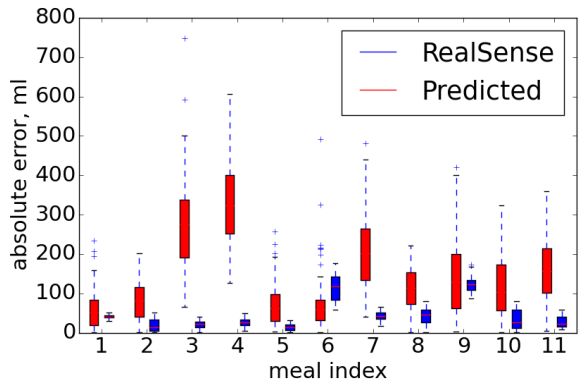



Figure 8. Absolute error in volume estimation (in ml) across the 11 meals in the NFood-3d dataset. Each meal has 100 images, taken from different viewing angles. The boxplots plot the distribution of errors across these 100 images.

dicted depth map (from CNN), and using the true segmentation mask (from human) or the predicted segmentation mask (from CNN). Unfortunately, we have found that our segmentation system does not work well on the NFood images, because their color and texture properties are too dissimilar to real food. However, we were able to compare the quality of using the true depth map and predicted depth map. The results are shown in Figure 8. We see that for most of the meals, our CNN volume predictor is quite accurate.

## 7. Calorie estimation

The final step is to map from the volume to the calorie content. This requires knowing the calorific density of each kind of food. The standard source for this is the USDA National Nutrient Database (NNDB). The latest version (May 2015) is Standard Release 27 [35], which lists nutritional facts about 8618 basic foods. However, we have noted a discrepancy of up to 35% in the calorie contents between the USDA NNDB and the numbers quoted by Nasco for their food replicas.[13]

A more serious problem is that the NNDB focuses on "raw" foods, rather than cooked meals. For example, NNDB contains information such as the calorie content of a pound of beef, but this does not help us estimate the calorie content of a cooked burger. For prepared foods, it is better to use the USDA Food and Nutrient Database for Dietary Studies (FNDDS) [16], which is derived from NNDB.[14] However, the calorie content can vary a lot depending on exactly how the food was prepared (e.g., grilling vs frying). Consequently, we do not yet have a broad coverage nutritional database for prepared foods, and therefore we have not yet been able to conduct an end-to-end test of our system outside of the restaurant setting.[15]

[13] According to http://www.enasco.com/product/WA29109HR, "nutrition data is provided by The Nutrition Company using their FoodWorks nutrient analysis software". Recall that these food replicas are used to train professional nutritionists, so the information cards that accompany them are designed to be as accurate as possible.

[14] For a detailed comparison of NDB-SR and FNDDS, see http://www.nutrientdataconf.org/PastConf/NDBC36/W-3_Montville_NNDC2012.pdf.

[15] Companies such as MyFitnessPal have developed much larger nutri-

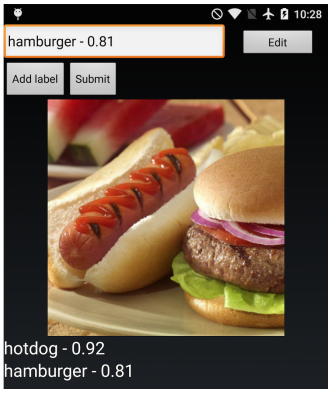



Figure 9. Screenshot of the mobile app. Note that it is running in airplane mode (no internet connection).

## 8. Mobile app

The complete system is sketched in Figure 1. We have ported the restaurant detector, meal detector, and food detectors (the multi-label classifiers) to the Android mobile phone system. The app can classify any image in under 1 second. The memory footprint for the model is less than 40MB (using unquantized floats to represent the CNN weights). However, we have not yet ported the segmentation or depth estimation CNNs.

To use the app, the user takes a photo and then our system processes it. First the system determines if the image contains a meal, and if you are at a known restaurant. If so, it applies the multilabel classifiers. We sort the possible labels by their predicted probability, taking all those above a threshold of 0.5, and then truncating to the top 5, if necessary. We then show these labels to the user (see Figure 9 for a screenshot). The user can dismiss any labels that are incorrect. He/ she can also enter new labels, either by selecting from a pull-down menu (sorted in order of probability), or by entering free text. The user's image and labels are then optionally stored in the cloud for subsequent offline model retraining (although we have not yet implemented this).

## 9. Related work

There is a large number of related prior publications. Here we mention a few of them.

In terms of public datasets, we have already mentioned Food101 [3], which has 101 classes and 101k images. In addition, there are various smaller datasets, such as: PFID [7], which has 61 classes (of fast food) and 1098 images; UNICT-FD889 [14], which has 889 classes and 3853 images; and UECFOOD-100 [24], which has 100 classes (of Japanese food), and 9060 images.

tional databases using crowd sourcing [22], but this data is proprietary, and its quality is not guaranteed (since most casual users will not know the true nutritional content of their food).

In terms of classifiers, [3, 18] use SVMs for generic foods. [2] and [1] develop restaurant-specific multi-label classifiers. Some recent papers on food segmentation include [28, 33, 38, 37].

There are many papers that leverage structure from motion to develop a 3d model of the food, including [28, 21, 10, 36, 32]. However, all these methods require multiple images and calibration markers. In terms of single images, [4] use parametric shape models for a small set of food types (e.g., sphere for an apple), and [19] use a nearest neighbor regression method to map the 2d image area to physical mass of each food item.

There are very few papers that predict calories from images. [33] apply semantic image segmentation, and then train a support vector regression to map from the number of pixels of each class to the overall number of calories in the meal. [4, 19] calculate calories by multiplying the estimated volume of each food by the calorie density. [26] use crowdsourcing to estimate calories. [1] relies on the restaurant's menu having the calorie information.

## 10. Discussion

Besides its obvious practical use, the Im2Calories problem is also very interesting from the point of view of computer vision research. In particular, it requires solving various problems, such as: fine-grained recognition (to distinguish subtly different forms of food); hierarchical label spaces (to handle related labels); open-world recognition (to handle an unbounded number of class names); visual attribute recognition (to distinguish fried vs baked, or with or without mayo); instance segmentation; instance counting; amodal completion of occluded shapes [20]; depth estimation from a single image; information fusion from multiple images in real time, on-device; etc. We have tackled some of these problems, but it is clear that there is much more work to do. Nevertheless, we believe that even a partial solution to these problems could be of great value to people.

## References

[1] O. Beijbom, N. Joshi, D. Morris, S. Saponas, and S. Khullar. Menu-match: restaurant-specific food logging from images. In *WACV*, 2015.

[2] V. Bettadapura, E. Thomaz, A. Parnami, G. D. Abowd, and I. Essa. Leveraging context to support automated food recognition in restaurants. In *WACV*, pages 580–587, 2015.

[3] L. Bossard, M. Guillaumin, and L. Van Gool. Food-101: Mining discriminative components with random forests. In *ECCV*, 2014.

[4] J. Chae, I. Woo, S. Kim, R. Maciejewski, F. Zhu, E. J. Delp, C. J. Boushey, and D. S. Ebert. Volume estimation using food specific shape templates in mobile image-based dietary assessment. In *Proc. SPIE*, 2011.

[5] C. Champagne, G. Bray, A. Kurtz, J. Montiero, E. Tucker, J. Voaufova, and J. Delany. Energy intake and energy expenditure: a controlled study comparing dietitians and non-dietitians. *J. Am. Diet. Assoc.*, 2002.

[6] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected CRFs. In *ICLR*, 2015.

[7] M. Chen, K. Dhingra, W. Wu, L. Yang, R. Sukthankar, and J. Yang. PFID: Pittsburgh fast-food image dataset. In *ICIP*, pages 289–292, 2009.

[8] F. Cordeiro, E. Bales, E. Cherry, and J. Fogarty. Rethinking the mobile food journal: Exploring opportunities for lightweight Photo-Based capture. In *CHI*, 2015.

[9] F. Cordeiro, D. Epstein, E. Thomaz, E. Bales, A. K. Jagannathan, G. D. Abowd, and J. Fogarty. Barriers and negative nudges: Exploring challenges in food journaling. In *CHI*, 2015.

[10] J. Dehais, S. Shevchik, P. Diem, and S. G. Mougiakakou. Food volume computation for self dietary assessment applications. In *13th IEEE Conf. on Bioinfo. and Bioeng.*, pages 1–4, Nov. 2013.

[11] J. Deng, J. Krause, A. C. Berg, and L. Fei-Fei. Hedging your bets: Optimizing accuracy-specificity trade-offs in large scale visual recognition. In *CVPR*, pages 3450–3457, June 2012.

[12] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common Multi-Scale convolutional architecture. *Arxiv*, 18 Nov. 2014.

[13] M. Everingham, S. M. Ali Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *IJCV*, 111(1):98–136, 25 June 2014.

[14] G. M. Farinella, D. Allegra, and F. Stanco. A benchmark dataset to study the representation of food images. In *ECCV Workshop Assistive CV*, 2014.

[15] FDA. www.fda.gov/Food/ IngredientsPackagingLabeling/LabelingNutrition/ucm248732.htm.

[16] USDA FNDDS. www.ars.usda.gov/ba/bhnrc/fsrg.

[17] B. Hariharan, P. Arbel, R. Girshick, and J. Malik. Simultaneous detection and segmentation. In *ECCV*, 2014.

[18] H. He, F. Kong, and J. Tan. DietCam: Multiview Food Recognition Using a MultiKernel SVM. *IEEE J. of Biomedical and Health Informatics*, 2015.

[19] Y. He, C. Xu, N. Khanna, C. J. Boushey, and E. J. Delp. Food image analysis: Segmentation, identification and weight estimation. In *ICME*, pages 1–6, July 2013.

[20] A. Kar, S. Tulsiani, J. Carreira, and J. Malik. Amodal completion and size constancy in natural scenes. In *Intl. Conf. on Computer Vision*, 2015.

[21] F. Kong and J. Tan. DietCam: Automatic dietary assessment with mobile camera phones. *Pervasive Mob. Comput.*, 8(1):147–163, Feb. 2012.

[22] R. Macmanus. How myfitnesspal became the king of diet trackers, 2015. readwrite.com/2015/02/23/trackers-myfitnesspal-excerpt.

[23] C. K. Martin, H. Han, S. M. Coulon, H. R. Allen, C. M. Champagne, and S. D. Anton. A novel method to remotely measure food intake of free-living individuals in real time: the remote food photography method. *British J. of Nutrition*, 101(03):446–456, 2009.

[24] Y. Matsuda, H. Hoashi, and K. Yanai. Recognition of Multiple-Food images by detecting candidate regions. In *ICME*, pages 25–30, July 2012.

[25] Mealsnap app. tracker.dailyburn.com/apps.

[26] J. Noronha, E. Hysen, H. Zhang, and K. Z. Gajos. PlateMate: Crowdsourcing nutrition analysis from food photographs. In *Proc. Symp. User interface software and tech.*, 2011.

[27] Google places API. https://developers.google.com/places/.

[28] M. Puri, Z. Zhu, Q. Yu, A. Divakaran, and H. Sawhney. Recognition and volume estimation of food intake using a mobile device. In *WACV*, pages 1–8, Dec. 2009.

[29] Rise app. https://www.rise.us/.

[30] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *arXiv:1409.0575*, 2014.

[31] D. Schoeller, L. Bandini, and W. Dietz. Inaccuracies in self-reported intake identified by comparison with the doubly labelled water method. *Can. J. Physiol. Pharm.*, 1990.

[32] T. Stutz, R. Dinic, M. Domhardt, and S. Ginzinger. Can mobile augmented reality systems assist in portion estimation? a user study. In *Intl. Symp. Mixed and Aug. Reality*, pages 51–57, 2014.

[33] K. Sudo, K. Murasaki, J. Shimamura, and Y. Taniguchi. Estimating nutritional value from food images based on semantic segmentation. In *CEA workshop*, pages 571–576, 13 Sept. 2014.

[34] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[35] USDA National Nutrient Database for Standard Reference, Release 27 (revised). http://www.ars.usda.gov/ba/bhnrc/ndl.

[36] C. Xu, N. Khanna, C. B. Y. He, and A. Parra. Image-Based food volume estimation. In *CAE workshop*, 2013.

[37] W. Zhang, Q. Yu, B. Siddiquie, A. Divakaran, and H. Sawhney. 'Snap-n-eat': Food recognition and nutrition estimation on a smartphone. *J. Diabetes Science and Technology*, 9(3):525–533, 2015.

[38] F. Zhu, M. Bosch, N. Khanna, C. J. Boushey, and E. J. Delp. Multiple hypotheses image segmentation and classification with application to dietary assessment. *IEEE J. of Biomedical and Health Informatics*, 19(1):377–388, Jan. 2015.

[39] F. Zhu, M. Bosch, I. Woo, S. Kim, C. J. Boushey, D. S. Ebert, and E. J. Delp. The use of mobile devices in aiding dietary assessment and evaluation. *IEEE J. Sel. Top. Signal Process.*, 4(4):756–766, Aug. 2010.

# Bibliography


[1] Dan Ciresan, Ueli Meier, Jonathan Masci, and Jürgen Schmidhuber. A committee of neural networks for traffic sign classification. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 1918–1921. IEEE, 2011.

[2] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 1701–1708. IEEE, 2014.

[3] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*, 2014.

[4] James J. Gibson. The theory of affordances. *Perceiving, Acting, and Knowing: Toward and Ecological Psychology*, pages 67–82, 1977.

[5] Michael Beetz, Ulrich Klank, Ingo Kresse, Alexis Maldonado, Lorenz Mösenlechner, Dejan Pangercic, Thomas Rühr, and Moritz Tenorth. Robotic roommates making pancakes. In *International Conference on Humanoid Robots*, pages 529–536. IEEE, 2011.

[6] Stefan Hinterstoisser, Stefan Holzer, Cedric Cagniart, Slobodan Ilic, Kurt Konolige, Nassir Navab, and Vincent Lepetit. Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes. In *Proc. International Conference on Computer Vision*, pages 858–865. IEEE, 2011.

[7] Louise Stark and Kevin Bowyer. Function-based generic recognition for multiple object categories. *CVGIP: Image Understanding*, 59(1):1–21, 1994.

[8] Ashutosh Saxena, Justin Driemeyer, and Andrew Y Ng. Robotic grasping of novel objects using vision. *The International Journal of Robotics Research*, 27(2):157–173, 2008.

[9] Michael Stark, Philipp Lies, Michael Zillich, Jeremy L. Wyatt, and Bernt Schiele. Functional object class detection based on learned affordance cues. In *International Conference on Computer Vision Systems (ICVS)*, May 2008.

[10] Jeannette Bohg and Danica Kragic. Grasping familiar objects using shape context. In *Int. Conf. on Advanced Robotics*, 2009.

[11] Charles C Kemp and Aaron Edsinger. Robot manipulation of human tools: Autonomous detection and control of task relevant features. In *Proc. of the Fifth Intl. Conference on Development and Learning*, 2006.

[12] Hedvig Kjellström, Javier Romero, and Danica Kragić. Visual object-action recognition: Inferring object affordances from human demonstration. *Computer Vision and Image Understanding*, 115(1):81–90, 2011.

[13] H. Grabner, J. Gall, and L. Van Gool. What makes a chair a chair? In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 1529–1536, 2011.

[14] V. Ferrari and A. Zisserman. Learning visual attributes. In *Advances in Neural Information Processing Systems*, 2007.

[15] Devi Parikh and Kristen Grauman. Relative attributes. *Proc. International Conference on Computer Vision*, 0:503–510, 2011.

[16] Christoph H. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 951–958, 2009.

[17] Xiaodong Yu and Yiannis Aloimonos. Attribute-based transfer learning for object categorization with zero/one training example. In *Proc. European Conference on Computer Vision*, 2010.

[18] Yuyin Sun, Liefeng Bo, and Dieter Fox. Attribute based object identification. In *International Conference on Robotics and Automation*, 2013.

[19] T. Hermans, F. Li, J. M. Rehg, and A. F. Bobick. Learning contact locations for pushing and orienting unknown objects. In *International Conference on Humanoid Robots*, 2013.

[20] Aitor Aldoma, Federico Tombari, and Markus Vincze. Supervised learning of hidden and non-hidden 0-order affordances and detection in real scenes. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1732–1739. IEEE, 2012.

[21] Hema Swetha Koppula, Rudhir Gupta, and Ashutosh Saxena. Learning human activities and object affordances from rgb-d videos. *International Journal of Robotics Research*, 32(8):951–970, 2013.

[22] Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Unsupervised feature learning for rgb-d based object recognition. In *International Symposium on Experimental Robotics*, 2012.

[23] Richard Socher, Brody Huval, Bharath Bhat, Christopher D. Manning, and Andrew Y. Ng. Convolutional-recursive deep learning for 3d object classification. In *Advances in Neural Information Processing Systems*, 2012.

[24] Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Multipath sparse coding using hierarchical matching pursuit. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2013.

[25] N. Silberman and R. Fergus. Indoor scene segmentation using a structured light sensor. In *Proceedings of the International Conference on Computer Vision - Workshop on 3D Representation and Recognition*, 2011.

[26] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *Proc. European Conference on Computer Vision*, 2012.

[27] Andrej Karpathy, Stephen Miller, and Li Fei-Fei. Object discovery in 3d scenes via shape analysis. In *International Conference on Robotics and Automation*, 2013.

[28] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and Sabine Susstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, 2012.

[29] John Lafferty, Andrew McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, 2001.

[30] Brian Fulkerson, Andrea Vedaldi, and Stefano Soatto. Class segmentation and object localization with superpixel neighborhoods. In *ICCV*, 2009.

[31] Y. Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2001.

[32] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *International Conference on Robotics and Automation*, 2011.

[33] Ian Lenz, Honglak Lee, and Ashutosh Saxena. Deep learning for detecting robotic grasps. *International Journal of Robotics Research (IJRR)*, 2014.

[34] Clément Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1915–1929, 2013.

[35] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[36] M.P. do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice-Hall, 1976.

[37] Jan J Koenderink and Andrea J van Doorn. Surface shape and curvature scales. *Image and vision computing*, 10(8):557–564, 1992.

[38] Tin Kam Ho. Random decision forests. In *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, volume 1, pages 278–282. IEEE, 1995.

[39] Antonio Criminisi and Jamie Shotton. *Decision forests for computer vision and medical image analysis*. Springer, 2013.

[40] Peter Kontschieder, Samuel Rota Bulo, Horst Bischof, and Marcello Pelillo. Structured class-labels in random forests for semantic image labelling. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2190–2197. IEEE, 2011.

[41] Piotr Dollár and C Lawrence Zitnick. Structured forests for fast edge detection. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 1841–1848. IEEE, 2013.

[42] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[43] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[44] R. Margolin, L. Zelnik-Manor, and A Tal. How to evaluate foreground maps? In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[45] Maurice George Kendall. *Rank correlation methods.* Griffin, 1948.

[46] Phil Brodatz. *Textures: a photographic album for artists and designers*. Dover Pubns, 1966.

[47] Matti Pietikäinen, Abdenour Hadid, Guoying Zhao, and Timo Ahonen. *Computer vision using local binary patterns*, volume 40. Springer Science & Business Media, 2011.

[48] Majid Mirmehdi, Xianghua Xie, and Jasjit Suri. *Handbook of texture analysis*. World Scientific, 2008.

[49] Thomas Leung and Jitendra Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *International journal of computer vision*, 43(1):29–44, 2001.

[50] Manik Varma and Andrew Zisserman. A statistical approach to texture classification from single images. *International Journal of Computer Vision*, 62(1-2):61–81, 2005.

[51] Manik Varma and Andrew Zisserman. A statistical approach to material classification using image patch exemplars. *IEEE transactions on pattern analysis and machine intelligence*, 31(11):2032–2047, 2009.

[52] Mario Fritz, Gary Bradski, Sergey Karayev, Trevor Darrell, and Michael J Black. An additive latent feature model for transparent object recognition. In *Advances in Neural Information Processing Systems*, pages 558–566, 2009.

[53] David A Forsyth and Margaret M Fleck. Automatic detection of human nudes. *International Journal of Computer Vision*, 32(1):63–77, 1999.

[54] Kristin J Dana, Bram Van Ginneken, Shree K Nayar, and Jan J Koenderink. Reflectance and texture of real-world surfaces. *ACM Transactions on Graphics (TOG)*, 18(1):1–34, 1999.

[55] Barbara Caputo, Eric Hayman, Mario Fritz, and Jan-Olof Eklundh. Classifying materials in the real world. *Image and Vision Computing*, 28(1):150–163, 2010.

[56] Lavanya Sharan, Ruth Rosenholtz, and Edward Adelson. Material perception: What can you see in a brief glance? *Journal of Vision*, 9(8):784–784, 2009.

[57] Ce Liu, Lavanya Sharan, Edward H Adelson, and Ruth Rosenholtz. Exploring features in a bayesian framework for material recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 239–246. IEEE, 2010.

[58] Diane Hu, Liefeng Bo, and Xiaofeng Ren. Toward robust material recognition for everyday objects. *Proceedings of the British Machine Vision Conference*, pages 48.1–48.11, 2011.

[59] Xianbiao Qi, Rong Xiao, Chun-Guang Li, Yu Qiao, Jun Guo, and Xiaoou Tang. Pairwise rotation invariant co-occurrence local binary pattern. *IEEE transactions on pattern analysis and machine intelligence*, 36(11):2199–2213, 2014.

[60] Sean Bell, Paul Upchurch, Noah Snavely, and Kavita Bala. Material recognition in the wild with the materials in context database. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 3479–3487, 2015.

[61] Sean Bell, Paul Upchurch, Noah Snavely, and Kavita Bala. Opensurfaces: A richly annotated catalog of surface appearance. *ACM Transactions on Graphics (TOG)*, 32(4):111, 2013.

[62] Shuai Zheng, Ming-Ming Cheng, Jonathan Warrell, Paul Sturgess, Vibhav Vineet, Carsten Rother, and Philip HS Torr. Dense semantic image segmentation with objects and attributes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3214–3221, 2014.