Active Grounding of Visual Situations

Max H. Quinn[1], **Erik Conser**[1], **Jordan M. Witte**[1], **Melanie Mitchell**[1,2]

[1]Portland State University

[2]Santa Fe Institute

Unpublished Draft

Abstract

We address a key problem for computer vision: retrieving images that are instances of *visual situations*. Visual situations are concepts such as "a boxing match", "a birthday party", "walking the dog", "a crowd waiting for a bus," "a handshake", or "a game of ping-pong," whose instantiations in images are linked more by their common spatial and semantic structure than by low-level visual similarity. While computer vision has made remarkable progress on recognizing individual objects in images, the problem of visual situation recognition is much more difficult for many reasons, including the vast variability of possible instances of a given situation, as well as the combinatorics of evaluating possible pairwise or multiple-object relationships. In this paper we describe a novel architecture we have developed for visual situation retrieval. Given a situation description, our architecture—called Situate—learns models capturing the visual features of expected objects as well as probabilistic spatial models capturing the expected spatial configuration of relationships among objects. Given a new image, Situate uses these models in an attempt to *ground* (i.e., to create a bounding box representing) each expected component of the situation in the image via an active search procedure. Situate uses the resulting grounding to compute a score indicating the degree to which the new image is judged to contain an instance of the situation. Such scores can be used to rank images in a collection as part of a retrieval system.

In the preliminary study described here, we demonstrate the promise of this system—and the importance of active grounding—by comparing Situate's retrieval and grounding performance on one example situation category with that of two baseline methods, as well as with a related image-retrieval system based on "scene graphs".

1. Introduction

The ability to automatically search for images with specified properties is a key topic for computer vision. In a world deluged with image data, automated image search and retrieval has become as important as text search, and progress in this area will have profound impacts thoughout society, in areas as diverse as medical diagnosis, public health, national security, privacy, and personal data organization.

Using deep neural networks, automatic detection of individual objects or specific faces in images has become remarkably successful [1, 2]. One can search an image collection for photographs containing, say, guitars, wine bottles, Golden Retrievers, or one's college roommate, and obtain many if not most of the relevant images with few false positives.

However, in many domains, users need to search for images with more complex or abstract properties, in which multiple objects with specified attributes are related in specific ways. Here are some examples: "a boxing match", "a birthday party", "a person walking a dog", "a crowd waiting for a bus," "a handshake", or "a game of ping-pong". Instances of such abstract visual concepts—which we call *visual situations*—are linked more by their common spatial and semantic structure than by low-level visual similarity or by the specific objects they contain. In general, automatically recognizing instances of a given visual situation is a difficult problem for several reasons, including substantial variability in visual features and spatial layout among different instances. Moreover, while

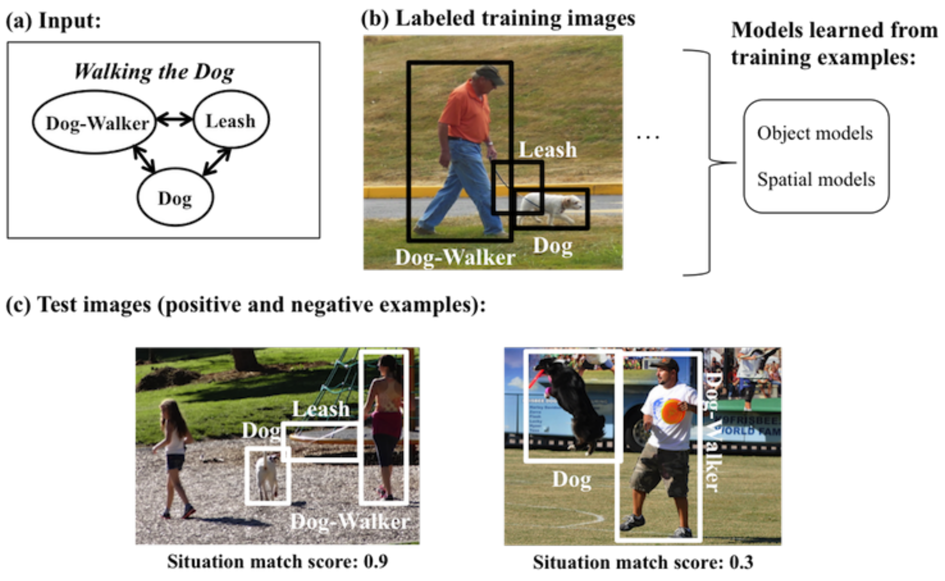
(a) Input:
Walking the Dog
Dog-Walker
Leash
Dog
(b) Labeled training images
Leash
Dog-Walker
Dog
...
Models learned from training examples:
Object models
Spatial models
(c) Test images (positive and negative examples):
Leash
Dog
Dog-Walker
Situation match score: 0.9
Dog
Dog-Walker
Situation match score: 0.3

Figure 1. Overview of Situate's training and testing pipeline, with the example situation "Walking the Dog". (a) The user specifies the object categories relevant to the given situation. This simple example includes three expected objects: (*Dog-Walker*, *Leash*, and *Dog*); the double-arrows indicate unnamed (i.e., to be learned) relationships among these objects. (b) Situate is also given a set of labeled training images, which include only positive examples of the situation, with relevant objects indicated via labeled bounding boxes. From these training examples, Situate learns models to recognize the individual objects ("object models"), as well as probabilistic models capturing the expected spatial configuration of the objects ("spatial models"). (c) Situate uses its learned models to score new ("test") images as instances of the given situation, by attempting to ground the expected object categories and their expected spatial configuration in the image. In this hypothetical example, the test image on the left is a good fit to the learned visual and spatial features of the situation so obtains a high match score, whereas the test image on the right, with no leash and an unexpected spatial relationship between *Dog-Walker* and *Dog*, is not a good fit, so obtains a low score. These steps will be described in more detail in Section 2. (This and other figures in this paper are best viewed in color.)

state-of-the-art object detection methods often rely on evaluating large numbers of "object proposals" at every location and scale of the image, the combinatorics of such exhaustive evaluation become much worse when the multiple objects, attributes, and possible relationships of a situation need to be considered. And while successful object detection has relied on huge amounts of labeled training data [3], there are few large labeled training sets for visual situations.

In this paper, we describe a novel architecture for retrieving instances of a query visual situation in a collection of images. Our architecture—called *Situate*—uses object-recognition models based on visual features, along with probabilistic models that represent learned multi-object relationships, in order to compute a rating of an image as an instance of the query situation. Situate learns these models from labeled training images; it applies these models to a new image via an active search process that attempts to ground components of the query situation in the image—that is, to create bounding boxes that localize relevant objects and relationships, and that ultimately provide a *situation match score* for the situation with respect to the image. The match scores can be used to rank the images in the collection with respect to the query situation; the highest ranking images can be returned to the user. Figure 1 illustrates Situate's training and testing pipeline.

We hypothesize that Situate's learned object and spatial models, used in tandem with its active situation-grounding method, will result in superior image retrieval performance than methods without these components. In this preliminary study, we test this hypothesis by running Situate on a challenging visual-situation dataset, and comparing its performance with two baseline methods: a "lesioned" version of Situate that lacks its spatial models and the feedback they provide the system, and an adapted version of the widely used Faster-RCNN object-detection method [2]. We also compare Situate's performance with that of a related semantic image retrieval system base don "scene graphs" [4]

2. Situate's Architecture

Situate's architecture is inspired by *active* approaches to perception, in which the perceiver acquires information dynamically, and in which the information acquired continually feeds back to control the perceptual process. In particular, for humans, recognizing a visual situation is an active process that unfolds over time, in which prior knowledge interacts with visual information as it is perceived, in order to guide subsequent eye movements. This interaction of top-down expectations and bottom-up perception enables a human viewer to very quickly locate relevant aspects of the situation [5, 6, 7, 8, 9]. A particular inspiration for Situate in the AI literature is the Copycat system of Hofstadter and Mitchell [10], which applies this kind of active high-level perception to the task of making analogies in an abstract non-visual domain.

As shown in Figure 1, a user's initial input to Situate is a specification of the object categories relevant to a particular situation, along with a training set—a collection of images representing positive instances of the situation, in which the relevant objects are labeled with bounding boxes.

2.1. Training

During the training process, Situate learns an *object model* for each relevant object category as well as an *object refinement model* for each category, described below. Situate also learns a *spatial model* that represents information about the expected spatial configuration of the relevant objects. Finally, the system learns a set of independent probability distributions ("priors") capturing the expected size and shape of each relevant object.

Object Models: The input to each category-specific object model is an *object proposal*, which specifies an object category along with set of coordinates specifying bounding box b in an image. For an object model corresponding to category C, the model's output is a prediction of the amount the overlap of b with a ground-truth object of category C. Following standard practice in the object-detection literature, overlap is measured as the *intersection over union* (IOU) of the proposed bounding box with a (human-labeled) ground-truth bounding box. (The IOU of two boxes measures the area of their intersection divided by the area of their union.) Figure 2 illustrates how object models work in our system. The figure shows two object proposals (white rectangles), along with human-created ground-truth boxes (black rectangles). The *Dog* proposal overlaps the ground-truth *Dog* box, and the *Dog* model predicts the overlap is 0.4 (here, an overestimate). The *Leash* proposal overlaps the ground-truth *Dog-Walker* box, and the *Leash* model correctly predicts zero overlap with a ground-truth *Leash* box.

Each object model is implemented as a linear combination of features obtained from running a pretrained convolutional network on the input region. We use the open-source

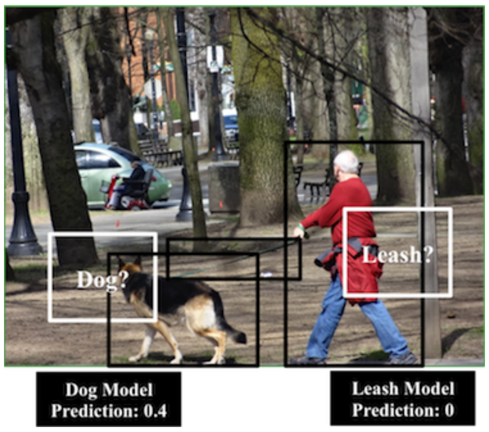

Figure 2. Illustration of learned object models. Each category-specific object model inputs an object proposal (specifying a bounding box) and outputs a prediction for the overlap of the proposal with a ground-truth bounding box for the given category. Two examples are shown. On the left, a *Dog* proposal (white box) is predicted to have 0.4 overlap with a *Dog* ground-truth box (where overlap is measured as *intersection over union* or IOU). On the right, a *Leash* proposal is predicted to have zero overlap with a *Leash* ground-truth box. The *Dog* proposal indeed overlaps a *Dog* ground-truth box (black box), though by less than 0.4. The *Leash* proposal overlaps a *Dog-Walker* ground-truth box but not a *Leash* box, so the zero IOU prediction is correct.

VGG-f network pre-trained on Imagenet [11] to obtain 4096 features from the fc7 layer, and ridge regression to learn the coefficients of the linear model. The ridge regression model we used [12] is trained on features extracted from training-image crops that partially or completely overlap ground-truth boxes.

Object Refinement Models: Situate similarly learns category-specific *object refinement models* from the same training crops used to learn object models. These refinement models—based on the "bounding-box regression" approach described in [13]—input an object proposal and output a new, "refined" object proposal that is predicted to have higher IOU with a ground-truth bounding box of the given category. Like the object models described above, each category-specific object refinement model is a linear combination of 4096 features from the pre-trained VGG fc7 layer (obtained by feeding the image crop defined by the object proposal through the VGG network). The refinement model inputs the original proposal's bounding box coordinates and outputs new bounding box coordinates.

As we will describe below, the refinement models will be applied to object proposals whose score from corresponding object models is above a pre-set threshold.

Spatial Model: For a given situation, Situate learns a *spatial model* that represents the expected spatial configuration among the relevant objects in that situation. The spatial model is a multivariate Gaussian distribution learned from human-labeled bounding boxes in training images. The variables in the distribution are bounding-box parameters—center coordinates, area ratio (i.e, area of box divided by area of image), and aspect ratio—from the relevant objects in a given situation. For example, for the *Walking the Dog* situation shown in Figure 1(a), the variables are the bounding box center coordinates, area ratios, and aspect ratios of the *Dog Walker*, *Dog*, and *Leash* boxes. As we will describe below, when Situate is run on a new image, once it has made a candidate detection of one or more relevant objects, it conditions the spatial model on those detections to narrow the expected location, shape, and size of the related objects.

We used a multivariate Gaussian due to its simplicity and fast speed of sampling, but plan to investigate using more sophisticated probabilistic models in future work.

Priors on Object Size and Shape: Situate also learns prior expectations of each relevant object category's size (area ratio) and shape (aspect ratio). These expectations are learned by fitting the area ratios and aspect ratios of ground-truth boxes as independent category-specific log-normal distributions. We used log-normal distributions to model these values rather than normal distributions, because the former are always positive and give more weight to smaller values. This made log-normal distributions a better fit for the data. Note that our system does not learn prior distributions over bounding-box *location*, since we do not want the system to model photographers' biases to put relevant objects near the center of the image.

2.2. Running Situate on a Test Image

After the models described above have been learned from training data, Situate is ready to run on new ("test") images. The input to Situate is an image and the program's output is (1) a *situation match score* that measures Situate's assessment of this image as an instance of the given situation, and (2) a set of "groundings"—detections of situation components in the Workspace, as was illustrated in Figure 1(c).

Following the *Copycat* architecture of Hofstadter and Mitchell [10], Situate produces these outputs by attempting to actively ground the situation components via the actions of *perceptual agents* in a *Workspace*. The agents are selected and run over a series of time steps, and create detections by combining bottom-up visual information with top-down expectations. The advantage of this active, temporal approach is that detections made in previous time steps can affect the actions of agents at subsequent time steps. We hypothesize that this interaction of continually updated top-down expectations and bottom-up perception will enable the system to quickly and reliably locate relevant aspects of the situation.

The detailed process by which Situate runs on a test image is illustrated in Figures 3 and 4, which show visualizations of eight time-slices from a run of the program using the *Walking the Dog* situation of Figure 1(a). We will explain Situate's approach via this example run.

Figure 3(a) shows Situate's state before any agents have run. The Workspace contains the unprocessed image. Situate's goal is to determine how well this image instantiates the query situation by attempting to locate the relevant objects that best fit Situate's learned models for this situation.

The gray squares shown below the Workspace represent the probability distributions over location for each object category. The uniform gray indicates that at this time step these are uniform distributions. Once an object is detected, these distributions will be updated to be conditioned on that detection according to the learned joint distribution. As we described above, the system also maintains probability distributions for aspect and area ratio of each object category (not shown here). Initially these "shape" and "size" distributions are the learned independent priors for each category, but once an object is detected these distributions will also be conditioned on that detection.

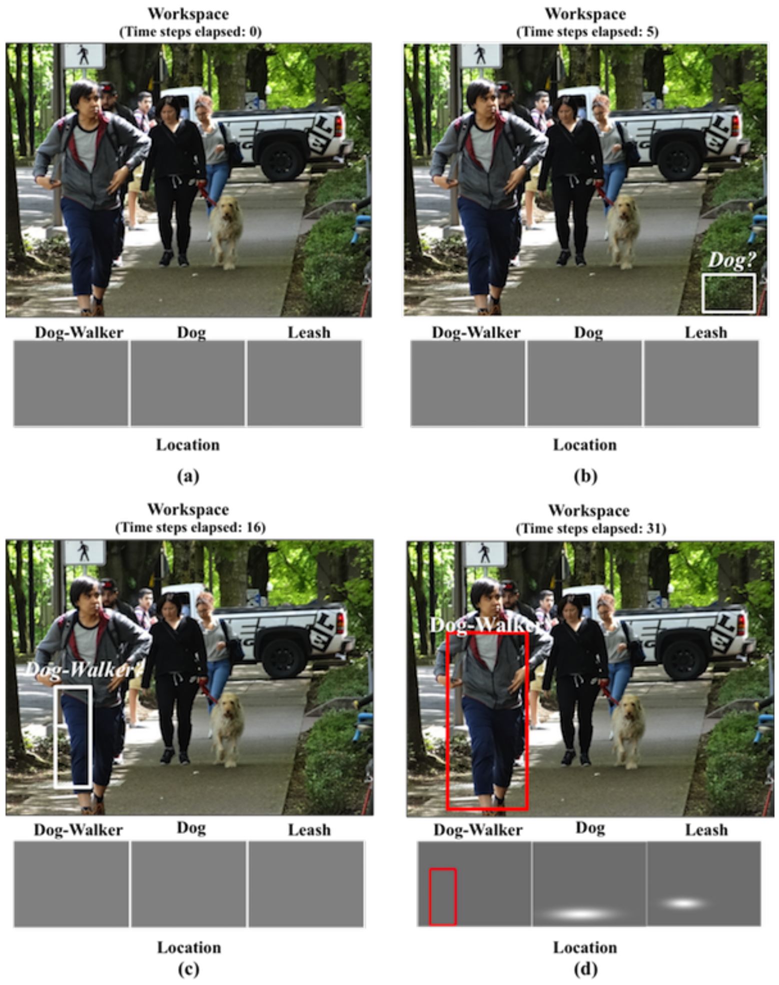

Figure 3. (a) **Time step 0:** State of Situate at beginning of a run on a test image. The Workspace contains the image that agents will act on. The three gray boxes below the Workspace show the current probability distributions over locations—initially all uniform—for the three relevant object categories. The distributions for aspect ratio (box shape) and area ratio (box size) for each category are initially set to the learned priors (not shown here). (b) **Time step 5:** At each time step, a single agent runs. Here, a *Dog* proposal (white box) has been created by a proposer agent by sampling from the current location, shape, and size distributions for *Dog*. The internal and external support of this proposal are below the threshold for follow-up, so the proposal will be discarded. (c) **Time step 16:** A *Dog-Walker* proposal is being considered. Its internal support is high enough to cause its proposer agent to post a follow-up *refiner agent* to the agent pool. (d) **Time step 31:** The refiner agent has improved the *Dog-Walker* proposal, and its support is high enough for its agent to create a detection (red box). The *Dog* and *Leash* location distributions are now conditioned on this detection.

At each time step, one agent is run. The system has two types of agents: *proposers* and *refiners*. A proposer chooses an object category from the list of relevant categories (here, *Dog-Walker*, *Dog*, and *Leash*), and samples from that category's current location, shape, and size distributions to create an object proposal. Figure 3(b) shows such a proposal, represented as a white box labeled "*Dog?*." The proposer agent run at time step 5 created this proposal; the agent evaluates the proposal via two measures: *internal* and *external* support. The internal support is a function of the *Dog* object model's prediction of overlap between this proposal and a ground-truth dog. The external support is a function that measures how well this proposal would fit in with other detections that have been made. These two measures are combined into a *total support* measure, which reflects the system's current judgement of the quality of this proposal for the given situation. If the internal support is above a pre-defined threshold, the proposal will be marked for possible

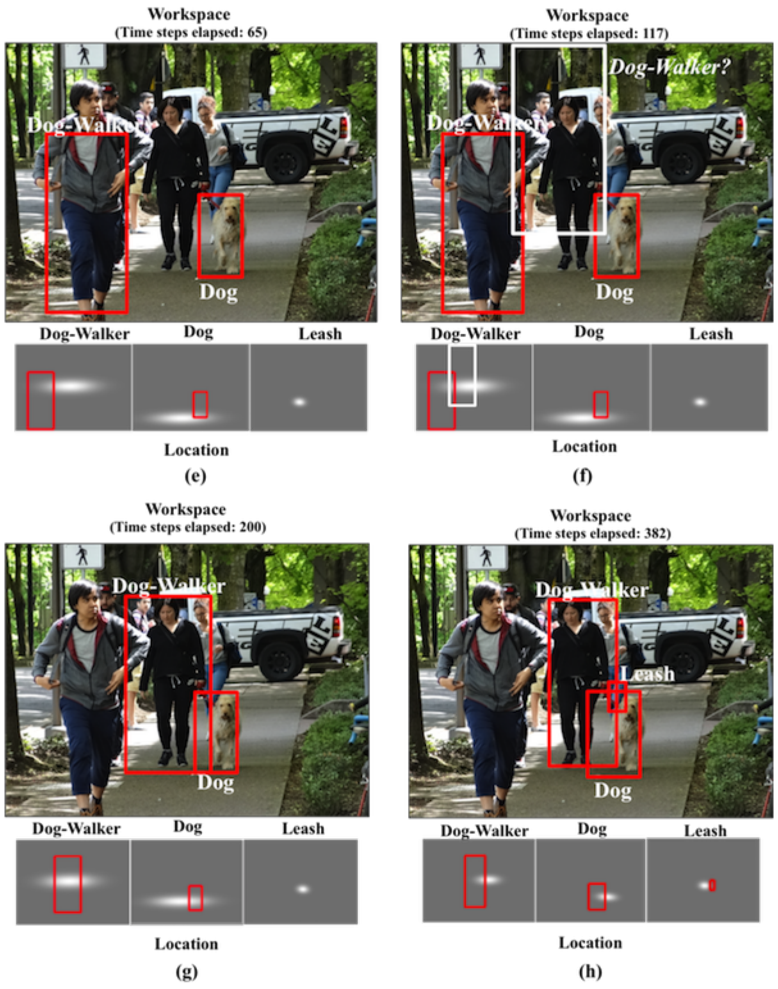

Figure 4. Continued from Figure 3. (e) **Time step 65:** A *Dog* detection has been made. The distributions for location, shape, and size of each object are now conditioned on the two detections in the Workspace. (f) **Time step 117:** An alternative proposal for *Dog-Walker* has been made. Due to the *Dog-Walker* location distribution conditioned on the *Dog* detection, this proposal has higher external support than the current *Dog-Walker* detection. (g) **Time step 200:** After refinement, the new *Dog-Walker* proposal has made into a detection, replacing the old *Dog-Walker* detection. The spatial distributions have been updated to be conditioned on the current detections in the Workspace. (h) **Time step 382:** With the help of support from the *Dog-Walker* and *Dog* detections, a *Leash* detection has been made, completing the grounding of the situation.

refinement; if the total support is above a second threshold, the proposal will be accepted as a *detection*. Since internal suppport is very low for the proposal in Figure 3(b), it will be discarded. (Due to space constraints in this paper, the specific forms of the internal, external, and total support functions, along with values of the thresholds we used, will be given in an online supplementary materials section when this paper is published.)

Figure 3(c) shows a *Dog-Walker* proposal at time step 16. This proposal's internal support is high enough for the system to create a refiner agent to try to improve the proposal.

When it runs, the refiner agent will use the learned object-refinement model to attempt to improve the original object proposal. As we described above, the refinement model inputs the original bounding-box coordinates and outputs new bounding box coordinates. The refiner agent creates an object proposal with the new bounding box and evalutes its internal and external support.

Figure 3(d) shows the result of this process. The refiner agent that resulted from Figure 3(c) ran at time step 31, and has improved the *Dog-Walker* bounding box sufficiently that its total support (here, based only on internal support) is enough for a *detection* to be created (red box). A detection is a data structure in the Workspace indicating that the system is confident that it has located a particular object. The spatial model is now conditioned on this detection, yielding new location distributions for *Dog* and *Leash*. (The size and shape distributions for these categories, not shown here, are also conditioned on the detection.) Agents will now increasingly focus on searching for objects in higher-probability areas of these distributions.

Note that 15 times steps have passed between a refiner agent being created (time step 16) and running (time step 31). Why doesn't it simply run immediately? This brings us to the question of *how* an agent is selected to run at each time step. The system maintains a fixed-size pool of agents, initially all proposer agents, with refiner agents added as object proposals are evaluated. At each time step, one agent is chosen at random from the pool. To avoid the pool being overrun with refiner agents, each refiner's probability of being chosen is less than that of the agent that created it. We found this regulation to be important in balancing the system's ability to explore new proposals versus refining existing proposals.

Considering Figure 3(d): The only problem is that the program has identified the wrong person as the dog-walker. As shown in Figure 4, this will be corrected in subsequent time steps using additional information discovered by the program.

At time step 65 (Figure 4(e)), a *Dog* proposal with sufficiently high support has resulted in a *Dog* detection in the Workspace. The location, shape, and size probability distributions have been updated to be jointly conditioned on the two detections (*Dog-Walker* and *Dog*). In particular, notice that the *Dog-Walker* location distribution has been conditioned on the *Dog* detection, and that the current *Dog-Walker* detection is offset from the center of that distribution. Similarly, the new *Dog* detection is offset from the center of the *Dog* location distribution. Even though both *Dog-Walker* and *Dog* have been detected, these (and likewise any detection) are treated as provisional detections until the end of the run. Agents will continue to search for better-fitting alternatives.

At time step 117, after agents have considered several such alternatives (and rejected them), an agent has created an alternative *Dog-Walker* proposal (Figure 4(f), white box) that is closer to the center of the current *Dog-Walker* distributions, and thus has more external support than the existing *Dog-Walker* detection. At time step 200 (Figure 4(g) a refiner agent has proposed an improved *Dog-Walker* box that has higher total support than the previous *Dog-Walker* detection; the old detection is deleted and the new one is created. The *Dog-Walker* and *Dog* boxes now both have high internal and external support (that is, they support each other very well according to the spatial model). The *Leash* location, size, and shape distributions, conditioned on both the *Dog-Walker* and *Dog* distributions, are now quite strongly peaked. This helps the system locate the small, hard-to-see leash (Figure 4(h)), at which point the run concludes with all of the situation objects having been detected. (If not all objects are detected, the run stops after a pre-set maximum number of time steps.) At the end of a run, the system computes the *situation match score* as a function of the total support of each detection. In the current version of

Situate, we define the situation match score as the geometric mean over the total support values of detections in the Workspace. If no detections are made by the end of the run, the situation match score is set to a minimum value—0.01 in the current version. We chose this "padded" geometric mean function as a simple way of combining total support scores, but will investigate alternative scoring methods in future work.

In summary, the following gives the main loop of Situate.

> **Input:** A test image
>
> **Initialization:** Initialize location, area-ratio, and aspect-ratio distributions for each relevant object category. The initial location distributions are uniform; initial area-ratio and aspect-ratio distributions are learned from training data. Initialize agent pool with proposer agents.
>
> **Main Loop:** Repeat until all relevant objects are detected or at most for *Max-Iterations*:
>
> 1. Choose agent from agent pool.
> 2. Run agent (and if agent is a proposer, replace it in the agent pool).
> 3. Update spatial model, conditioned on current detections in the Workspace.
>
> **Return:** Situation match score S. where
>
> $$S = \max\left[0.01, \left(\prod_{i=1}^{n} \text{total-support}(d_i)\right)^{\frac{1}{n}}\right], \tag{1}$$
>
> where n is the number of detections in the Workspace, and d_i is the ith detection. For the experiments described in this paper, we used *Max-Iterations* = 1,000.

In designing Situate's architecture, we were inspired by Hofstadter et al.'s idea of modeling perception as a "parallel terraced scan" [10], in which many possible exploration paths are pursued in parallel, but not all at the same speed or to the same depth. Moreover, the exploration is "active" in the sense that information is used as it is gained to continually modify the resources given to possible paths. Like the *codelets* in the architecture of [10], our (serially run) architecture approximates such a parallel search strategy by interleaving many independent agents. In principle, many of these explorations could be performed in parallel. Furthermore, splitting up "proposers" and "refiners" allows the system to balance time spent on bottom-up exploration with that on focused follow-ups.

An advantage of such an approach is balancing the need to explore many possibilities while still avoiding exhaustive evaluation of possible situation configurations. However, one possible problem with our approach is that the system can spend too much time considering lower quality possibilities (e.g., an incorrect dog-walker) on the basis of incomplete information. Hofstadter et al.'s architecture [10] included a *temperature* variable to help address this problem; temperature simultaneously measured the program's perceived quality of its interpretation at a particular time step and fed back to control the amount of randomness in the system. The lower the perceived quality, the higher the temperature, and the more randomness was injected into the system. This prevented the system from spending too much time investigating low-quality exploration paths. While we believe that such a mechanism is essential to successfully regulating this kind of architecture (see [14] for a discussion of analogous mechanisms in biological systems), implementing a similar temperature mechanism in Situate is a topic for future work. In the current system we employed a simple mechanism for injecting randomness: each proposer agent decides with probability $\epsilon \in [0, 1]$ to sample from a uniform location distribution rather than the current location distribution for its object category. In the experiments described below, we used $\epsilon = 0.5$.

We hypothesize that the approach we have described above will have superior performance on grounding elements of situations, and thus on ranking images, than methods that do not use this kind of active approach, assuming the same amount of training data. We also hypothesize that our method will be able to achieve this performance with significantly fewer object-proposal evaluations than non-active methods.

In the next section we review related work. This is followed by a description of the experiments we performed to test our hypotheses, and a discussion of the results. We conclude with an assessment of our hypotheses based on these initial experiments, and plans for future work.

3. Related Work

Our work falls under the broad area of "image understanding," which has a vast literature in AI. Here we describe some of the recent approaches most closely related to Situate's goals and architecture.

A recent approach to semantic image retrieval uses *scene graphs* to represent images and to query image collections. A scene graph is a graphical representation of objects, attributes, and relationships that encode an image or image region, or desired image content—e.g., "a tall man wearing a white baseball cap." Johnson et al. [4] developed an architecture for semantic image retrieval via "scene graph grounding" which uses a learned graphical model to determine the most likely set of bounding boxes and relationships between them that ground an input scene graph in a given image. In contrast to our focus on visual situations, Johnson et al. tested this approach both on highly detailed scene graphs representing a single target image, as well as on simple scene graphs referring to a pair of objects (e.g., the man and the hat in the example above). In addition, Johnson et al.'s approach differs from Situate in that, rather than taking an active, dynamic approach to locating objects and relationships, it exhaustively considers a large set of bounding box proposals and all pairwise relationships amoug those proposals. We will describe Johnson et al.'s approach in more detail in Section 5, and describe our results comparing its performance to that of Situate. A different approach to image retrieval via scene graphs was proposed in [15]: their system embedded both the query scene graph and test image in a common space and computed a match score. Several groups have focused on automatically creating scene graphs from image data (e.g., [16, 17, 18]).

Another widely studied image-understanding task is that of "grounding referential expressions." Given a phrase such as "the brown dog next to the woman wearing sunglasses," the task is to locate the object being referred to, by grounding each object and relation in the phrase. Examples of recent work in this area include [19, 20, 21]. Like the scene-graph task described above, research on this task has focused on specific "free-form" phrases rather than more abstract situation descriptions—the open-ended nature of the task makes it very difficult, and accuracies reported on large datasets have remained low to date. A related task is that of detecting visual relationships in images (e.g., [22]); to our knowledge, the literature on this task has focused almost exclusively on pairwise relationships (e.g. "dog riding surfboard"), rather than multi-object visual situations.

Our situation-retrieval task, like the scene-graph and expression-grounding tasks described above, shares motivation but contrasts with the well-known tasks of "event recognition" or '"action recognition" in still images (e.g., [23, 24, 25]). These latter tasks consist of classifying images into one of several event or action categories, without the requirement of localizing objects or relationships. A related task, dubbed "Situation Recognition" by [26], requires a system to, given an image, predict the most salient verb, along with its subject and object ("semantic roles" [27]). (It should be noted that we use the term "visual situation" in a more general sense.)

Our task also contrasts with recent work on automatic caption generation for images (e.g., [28, 29]), in which image content is statistically associated with a language generator. The goal of caption-generation systems is to generate a description of *any* input image. Even the versions with "attention" (e.g., [30]), which are able to highlight diffuse areas corresponding roughly to relevant objects, are not able to recognize and locate all important objects, relationships, and actions, or more generally to recognize abstract situations.

While the literature cited above does not include "active" detection methods such as Situate that involve feedback, there has been considerable work on active object detection (e.g., [31, 32]), often in the context of active perception in robots [33] and modeling visual attention [34, 35, 36]. More recently, several groups have framed active object detection as a Markov decision process and use reinforcement learning to learn a search policy (e.g., Caicedo2015).

This section has given a sampling (out of a large literature) of recent work on image understanding. While related in motivation and approach to some of these efforts, the specific problem we are addressing (visual situation retrieval) and method (active grounding of situation components) is, to our knowledge, unique in the literature.

4. Datasets

The computer vision community has created several important benchmark datasets for object recognition and detection (e.g., [37, 3]) and for some of the other tasks described in the previous section that combine vision and language (e.g., [38, 39]). None of these precisely offers the kind of data that we needed for our situation-retrieval task—that is, collections of numerous instances of specific multiobject situations, in which the objects are localized with ground-truth bounding boxes. (For example, the ImSitu "Situation Recognition" dataset is organized around verbs such as *carrying*, *jumping*, and *attacking*, each with one subject (e.g., "dog jumping") and some with one additional object that the subject acts upon (e.g., "man carrying baby").)

For our preliminary work with Situate, we developed a new dataset representing the "Walking the Dog" situation. We chose this situation category because it is reasonably easy to find sufficient varied instances to train and test our system, and these instances offer a variety of interesting challenges. This dataset, the "Portland State Dog-Walking Images," contains 500 positive instances of the *Dog-Walking* situation. The positive instances are photographs taken by members of our group, and in each we labeled (with bounding boxes) the *dog*, *dog walker*, and *leash*. Each image contained only one of each target object, but many also contained additional (non-dog-walking) people, along with cars, buildings, trees, and other "clutter". The challenges of this dataset include determining which person is the dog-walker, as well as locating dogs (often small, and sometimes partially occluded) and leashes (which are very often difficult, based on visual features alone, to distinguish from other line-like structures in an image), and deciding if the configuration of these three objects fits the learned dog-walking situation. For the experiments described below, we split the 500 images into a 400-image training set and a 100-image test set.

We also created a negative set of 400 images selected from the Visual Genome dataset [38]. This set includes images in which people interact with dogs in non-dog-walking situations, along with images with people but no dogs, dogs but no people, and neither. The positive and negative images we used can be downloaded from our project website [40].

Note that our collection is similar to the Stanford 40 Actions [41] "Walking the Dog" category, but the photographs in our set are more numerous, varied, and have bounding box labels for each relevant object. As we describe below, we also used a subset (145 images) of the Stanford "Walking the Dog" images—those with exactly one dog and one leash—as a second set of test examples. This subset can be also be downloaded from our project website [40].

In future work we will extend Situate to be able to group objects so as to be able to deal with multiple instances of particular object categories (e.g., "taking multiple dogs for a walk").

5. Methods

We performed experiments to evaluate Situate's image retrieval and situation grounding abilities. We also assess the importance of Situate's learned spatial models by comparing with two baseline methods that only use object appearance models (i.e., they do not use spatial models). Finally, we compare Situate's performance with that of the Image Retrieval using Scene Graphs (IRSG) method of Johnson et al. [4]. In each method, we performed any necessary training using the same training set that we used for Situate.

5.1. Baseline Methods

The first baseline, which we call the "Uniform Sampling" method, is identical to Situate except that agents always sample locations and bounding box parameters uniformly rather than using a learned spatial model. When creating an object proposal, an agent chooses a center location by sampling uniformly across the entire image, and chooses area and aspect ratios by sampling uniformly over fixed ranges.

The second baseline is based on the widely-used Faster-RCNN algorithm for object detection [2]. Faster-RCNN is a deep convolutional network that can be trained to propose bounding boxes and score them with repsect to given object categories; it has been shown to achieve state-of-the-art performance on object detection. We used an open-source version of Faster-RCNN [42] that was pre-trained on the Pascal VOC dataset, and we finetuned it for the *Dog-Walker*, *Dog*, and *Leash* categories using the same training data given to Situate. We then ran the finetuned Faster-RCNN network on each test image (positive and negative), and selected the highest scoring bounding box (as scored by Faster-RCNN) corresponding to each of the relevant object categories. (Following [2], we applied non-maximum suppression to boxes before selecting the highest scoring box.) Analogous to Situate, we defined the *Situation Grounding Score* as the padded geometric mean of the scores assigned to these bounding boxes by Faster-RCNN. Our goal was to see how well this specially trained Faster-RCNN model could be used to perform situation retrieval even though it does not use any situtation model.

5.2. IRSG Method

We also compare Situate's performance that of Johnson et al.'s "Image Retrieval using Scene Graphs" (IRSG) method [4]. IRSG is similar to Situate in that it scores an input image as to how well it instantiates a query description (represented as a scene graph), and uses image scores to rank images in a collection, with the goal of image retrieval. Moreoever, IRSG computes its score by attempting to ground components of the query scene graph in the input image.

IRSG first creates a set of object proposals that are not category specific—it does this using the geodesic object proposal method of [43]. IRSG then uses R-CNN [13] to give each object proposal multiple "appearance" scores—one for each object category in the scene graph. IRSG also uses a Gaussian mixture model (GMM), learned from training examples, which, given pairs of candidate bounding boxes, returns probabities of different possible object relationships . The "unary" object appearance scores and the binary relationship scores are used in a conditional random field model defined over a factor graph representing the query. As a simple example, for the query "man wearing hat next to woman," the system would exhaustively consider all possible pairs of boxes in each of the two relationships ("wearing" and "next to"), and see which configuration minimizes the energy function defined by the conditional random field.

We obtained the source code for IRSG from the authors of [4] and adapted it in order to compare it with Situate and our other methods. Instead of geodesic object proposals scored by R-CNN, we used the top-scoring 300 boxes per category (*Dog-Walker*, *Dog*, *Leash*) from our finetuned version of Faster-RCNN (following [2], we applied per-category non-maximal suppression to obtain these boxes). Since IRSG is (as currently implemented) limited to pairwise relationships, we trained the GMM on the spatial relationship between Dog-Walker and Leash, and between Leash and Dog (one could perhaps formulate this as the scene "Dog-Walker holding Leash and Leash attached to Dog"). The factor graph and conditional random field formulation and energy minimization was performed using the same algorithms that were described in [4]. In this way, each positive and negative test image in our set was scored (using the final energy value).

As we mentioned above, due to space constraints, we omit some details of Situate (e.g., the detailed forms of the external and total support functions, thresholds for detections) and the other methods. We will provide these details in an online supplementary materials section, along with our code and all data, when this paper is published.

5.3. Evaluation Metric

We ran each method on the three test sets: Portland Dog-Walking (100 images), Stanford Dog-Walking (145 images), and the 400 negative (non-dog-walking) examples chosen from the Visual Genome dataset. We use the Stanford images as a second positive test set to help explore the generality of Situate's learned models, since these images are from different sources than those in the Portland set: the former were collected by Stanford researchers using online image search engines, while the latter consists of photographs taken by members of our research group.

Our evaluation metric is Single-Image Recall $@N$ (abbreviated $R@N$). This measures the probability that, if a single positive example were added to the set of negative examples, and the collection was ranked by situation match score, the positive example would be in the N top-ranked images. For example, given our 100 positive and 400 negative test images, $R@10 = .57$ means that 57 out of 100 of the positive images would be in the top 10 ranked images if they were ranked alone with the 400 negative images.

The Situate and Uniform Sampling methods are stochastic, whereas Faster-RCNN and our adapted version of IRSG are deterministic. Below, for clarity of presentation, we give results from typical runs of Situate and Uniform Sampling, but note that runs using different random seeds produce very similar results with respect to the evaluation metric we use.

6. Results

Table 1 gives the Single-Image Recall $@N$ resulting from running the four different methods on the two positive test sets: Portland State Dog-Walking (top table) and Stanford Dog-Walking (bottom table) and the negative set (the same negative set of 400 images was used for each positive set). The best result in each row is boldfaced. It can be seen that Situate produced the best result on all but two rows: $R@100$ for the Portland images and and $R@1$ for the Stanford images. In many cases, Situate's Single-Image Recall $@N$ is quite a bit higher than that of the other methods, giving evidence for our hypothesis that Situate's active grounding method, together with its learned models, will result in superior image retrieval performance than methods lacking these components.

7. Discussion

What accounts for Situate's generally superior performance as compared with the other methods we tested? What kinds of errors does each method make? We investigated these

Portland State Dog-Walking Images

N	Situate	Uniform	Faster-RCNN	IRSG
1	**0.27**	0	0.24	0.13
5	**0.53**	0.16	0.28	0.21
10	**0.57**	0.30	0.38	0.40
20	**0.66**	0.41	0.54	0.57
100	0.89	0.74	0.91	**0.94**

Stanford Dog-Walking Dataset

N	Situate	Uniform	Faster-RCNN	IRSG
1	0.21	0	**0.28**	0.03
5	**0.54**	0.10	0.31	0.09
10	**0.60**	0.28	0.37	0.20
20	**0.72**	0.34	0.51	0.30
100	**0.93**	0.77	0.91	0.77

Table 1. Results for Single Image Recall@N. The top table gives results for the Portland State Dog-Walking test set (100 images) and the bottom table gives results for the Stanford Dog-Walking test set (100 images). Four methods (Situate, Uniform, Faster-RCNN, and IRSG) were run on the positive test sets as well as the negative examples. Each $R@N$ value is the probability that a single image from the test set would be in the top N ranked images when grouped with the 400 negative images. The best result in each row is boldfaced.

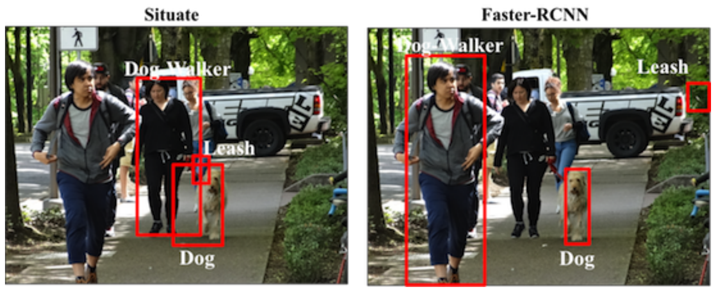

Figure 5. Final detections by Situate and by Faster-RCNN on the same test image, illustrating how Situate's context-aided detections resulted in a correct situation grounding, missed by Faster-RCNN.

questions by viewing the "situation groundings"—that is, the object detections produced by each method by the end of a run on each test image.

In the positive test images, we found that, while all four methods were good at detecting dog-walkers and dogs, leashes were often not detected or falsely detected in an incorrect location. The Uniform-Sampling method was by far the worst at detecting leashes, missing them entirely in about 80% of the test images. This was largely responsible for its poor performance.

Faster-RCNN was able to detect leashes with about the same success rate as Situate, but Situate was considerably better at detecting them together with Dog-Walkers and Dogs—that is, as parts of coherent situations. Figure 5 contrasts Situate's detections on the image from Figure 3 with Faster-RCNN's detections. As we discussed in Section 2, Situate was able to correct its initial dog-walker detection with the help of context from its dog detection, after which it used context from the two detections to locate the hard-to-see leash. These mechanisms are absent from Faster-RCNN, which was not able to locate the correct dog-walker or leash. (Faster-RCNN's detections gave a low score to this image, due to the low-scoring leash detection.)

This kind of context-aided detection accounted for part of Situate's superior $R@N$ perfor-

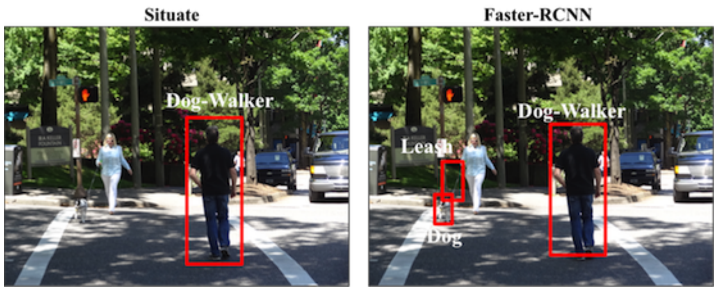

Figure 6. Final detections by Situate and by Faster-RCNN on the same test image, illustrating a case where Situate gets stuck by identifying an incorrect dog-walker, and is not able to locate the other objects, resulting in a low score for this image. In contrast, Faster-RCNN is able to locate the dog and leash (and gives a high score to this image), even though it has also detected the same incorrect dog-walker.

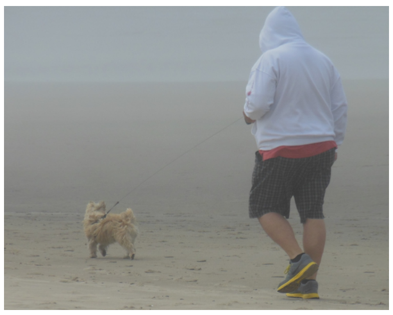

Figure 7. An example of an object that none of the methods was able to detect: the leash in this misty beach scene.

mance. Another factor was that when Faster-RCNN did make a correct leash detection, it often gave the detection a low score, since leashes are often only barely visible in these images. Situate was able to assign higher scores since its detection scores incorporated *external support*—that is, support from the context of the other detected objects. Faster-RCNN has no mechanism for doing this.

There were some cases where Situate failed and Faster-RCNN succeeded, such as the example in Figure 6. This was an example of Situate identifying the wrong person as the dog-walker, and being unable to escape from the resulting incorrect spatial probabilities for the other objects. Faster-RCNN, on the other hand, made the same incorrect dog-walker identification but was not constrained by this error; it succeeded in locating the dog and leash.

Finally, in the positive test set there were cases where all methods failed to locate an object, such as the leash in the misty beach scene of Figure 7. Apparently the object appearance models for *Leash* scored this leash very low.

In examining the results on the negative test images, we were able to see that all the methods were susceptible to false positives—high-scoring but incorrect detections. However, it seems that Faster-RCNN was more likely to make false-positive detections than Situate, since Situate's ability to dynamically perceive and use context prevented some potential false-positives from scoring highly.

The analysis we described in this section highlights one of the advantages of requiring the system to *ground* situation elements as part of the image-ranking process. In much

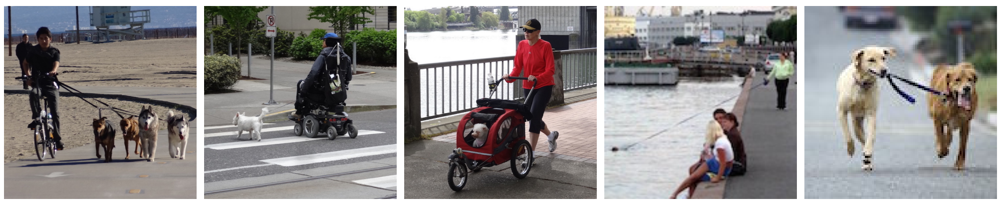

Figure 8. Five "non-prototypical" *Dog-Walking* situations that require conceptual slippage. (Images 1–3: Portland State Dog-Walking Images. Image 4: http://www.drollnation.com/gallery/2015/12/randomness-120315-5.jpg. Image 5: http://www.delcopetcare.com/wp-content/uploads/2013/02/dog-walking.jpg.)

of the existing scene categorization work in the literature, such explicit grounding is not performed; without it, it is often hard to determine *why* a computer vision system makes the categorizations—or errors—reported in the results. Here we were able to make sense of why one method exhibits superior performance to others, and what types of errors are made by the various methods.

[Note: Final draft will include analysis of ISRG detections.]

8. Conclusions and Future Work

In this paper we have described a preliminary study of Situate, a novel approach to visual situation retrieval. The results of this study have shown the promise of Situate's active situation-grounding architecture: our system's image-retrieval performance on the "Walking the Dog" situation generally surpassed that of two baselines as well as a related image-retrieval system from the recent literature. We showed how Situate is able to use information as it is gained in order to focus its search, and to use the support of context in order to locate hard-to-detect objects (e.g., barely visiable leashes, small dogs, partially occulded objects). In analyzing these results, we were able to understand some of the reasons for Situate's superior performance, as well as to identify some of its problems. This analysis underscores the important role of *grounding* situation elements as part of scoring an image.

Visual situation recognition and retrieval is a broad and difficult open problem in computer vision research, and the results we have presented highlight many avenues of future research. In the near term we plan to improve our algorithm in several ways: incorporating a fast-to-compute salience measure to provide location prior probabilities to agents; expanding the kind of object attributes that can be detected by agents (e.g., orientation and other pose features); expanding the types of relationships that can be identified (e.g., recognizing that two objects have the same orientation). We also plan to experiment with more sophisticated spatial probability models, such as Gaussian mixture models, while keeping in mind the tradeoff between sophistication and speed of computation.

Most importantly, we will explore the ability of our algorithms to scale to larger datasets and to generalize to other situation categories.

In the longer term, we will focus on, among other extensions, being able to speed up our active search method via parallelization. Finally, one of our original motivations for this project was to create a system that can recognize visual *analogies*. For example, most people would consider the images in Figure 8 to be (somewhat stretched) instances of the abstract *Dog-Walking* situation. This kind of recognition requires what Hofstadter and colleagues have called "conceptual slippage", in which the roles defining a situation (e.g., *Dog-Walker*) can be fluidly filled by concepts semantically related to the protogype (e.g., a "dog-walker" can be a person riding a bicycle, or driving a car, or even another dog). Making appropriate conceptual slippages is at the heart of analogy-making, which itself is a core aspect of cognition [44].

The abilities of computer vision remain far from human-level visual understanding, but we believe that progress on the problem of situation-recognition, particularly incorporating analogy-making, will play a pivotal role in giving computers the ability to make sense of what they see.

9. Acknowledgments

We are grateful to Justin Johnson of Stanford University for sharing the source code for the IRSG project, and to NVIDIA corporation for donation of a GPU used for this work. Many thanks to Garrett Kenyon, Bryan Lee, Sheng Lundquist, Anthony Rhodes, Evan Roche, Rory Soiffer, and Robin Tan for discussions and assistance concerning this project. This material is based upon work supported by the National Science Foundation under Grant Number IIS-1423651. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] H. Jiang and E. Learned-Miller, "Face detection with the faster R-CNN," in *2017 12th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2017)*. IEEE, may 2017, pp. 650–657.

[2] J. Ren, S., He, K., Girshic, R., Sun, "Faster R-CNN: Towards real-time object detection with region proposals," in *Advances in Neural Information Processing Systems 28 (NIPS 2015)*, 2015.

[3] R. Socher, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009, pp. 248–255.

[4] J. Johnson, A. Karpathy, and L. Fei-Fei, "DenseCap: Fully Convolutional Localization Networks for Dense Captioning," pp. 4565–4574, 2015.

[5] M. Bar, "Visual objects in context." *Nature Reviews Neuroscience*, vol. 5, no. 8, pp. 617–629, 2004. [Online]. Available: http://dx.doi.org/10.1038/nrn1476

[6] P. G. Malcolm, George L. and Schyns, "More than meets the eye: The active selection of diagnostic information across spatial locations and scales during scene categorization," in *Scene Vision: Making Sense of What We See*, M. Kveraga, K. and Bar, Ed. MIT Press, 2014, pp. 27–44.

[7] M. Neider and G. Zelinsky, "Scene context guides eye movements during visual search," *Vision Research*, vol. 46, no. 5, pp. 614–621, 2006.

[8] M. C. Potter, "Meaning in visual search," *Science*, vol. 187, pp. 965–966, 1975.

[9] C. Summerfield and T. Egner, "Expectation (and attention) in visual cognition." *Trends in Cognitive Sciences*, vol. 13, no. 9, pp. 403–9, 2009.

[10] D. R. Hofstadter and M. Mitchell, "The Copycat project: A model of mental fluidity and analogy-making," in *Advances in Connectionist and Neural Computation Theory*, K. Holyoak and J. Barnden, Eds. Ablex Publishing Corporation, 1994, vol. 2, pp. 31–112.

[11] "Imagenet-VGG-f." [Online]. Available: http://www.vlfeat.org/matconvnet/pretrained/

[12] “Matlab Ridge Regression.” [Online]. Available: https://www.mathworks.com/help/stats/ridge.html

[13] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 580–587.

[14] M. Mitchell, “Self-Awareness and Control in Decentralized Systems,” in *Working Papers of the AAAI 2005 Spring Symposium on Metacognition in Computation*. AAAI Press, 2005.

[15] R. Belilovsky, E., Blaschko, M. B., Kiros, J. R., Urtasun, R., Zemel, “Joint embeddings of scene graphs and images,” in *International Conference on Learning Representations*, 2017.

[16] S. Aditya, C. Baral, Y. Yang, Y. Aloimonos, and C. Fermuller, “DeepIU: An Architecture for Image Understanding,” *Advances in Cognitive Systems*, vol. 4, pp. 1–16, 2016.

[17] D. Xu, Y. Zhu, C. B. Choy, and L. Fei-Fei, “Scene Graph Generation by Iterative Message Passing,” *arXiv:1701.02426.*, 2017.

[18] M. Y. Yang, W. Liao, H. Ackermann, and B. Rosenhahn, “On support relations and semantic scene graphs,” *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 131, pp. 15–25, sep 2017.

[19] J. Mao, J. Huang, A. Toshev, O. Camburu, A. Yuille, and K. Murphy, “Generation and Comprehension of Unambiguous Object Descriptions,” *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11–20, 2016.

[20] V. K. Nagaraja, V. I. Morariu, and L. S. Davis, “Modeling context between objects for referring expression understanding,” in *Proceedings of the European Conference on Computer Vision*, vol. 9908 LNCS, 2016, pp. 792–807.

[21] A. Rohrbach, M. Rohrbach, R. Hu, T. Darrell, and B. Schiele, “Grounding of Textual Phrases in Images by Reconstruction,” in *Proceedings of the European Conference on Computer Vision*, 2016, pp. 817–834.

[22] C. Lu, R. Krishna, M. Bernstein, and L. Fei-Fei, “Visual relationship detection with language priors,” in *Proceedings of the European Conference on Computer Vision*, 2016, pp. 852–869.

[23] G. Guo and A. Lai, “A survey on still image based human action recognition,” *Pattern Recognition*, vol. 47, no. 10, pp. 3343–3361, 2014.

[24] L.-J. Li and L. Li Fei-Fei, “What, where and who? Classifying events by scene and object recognition,” in *International Conference on Computer Vision (ICCV)*. IEEE, 2007, pp. 1–8.

[25] L. Wang, Z. Zhe Wang, W. Wenbin Du, and Y. Qiao, “Object-scene convolutional neural networks for event recognition in images,” in *Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. IEEE, 2015, pp. 30–35.

[26] A. Yatskar, M, Zettlemoyer, L., Farhadi, “Situation recognition: Visual semantic role labeling for image understanding,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016.

[27] S. Gupta and J. Malik, “Visual Semantic Role Labeling,” *arXiv:1505.04474*, 2015.

[28] O. Vinyals, A. Toshev, and S. Bengio, “Show and Tell: Lessons learned from the 2015 MSCOCO Image Captioning Challenge,” *IEEE Transactions on*, 2015.

[29] J. Johnson, R. Krishna, M. Stark, L.-J. Li, D. Shamma, M. Bernstein, and L. Fei-Fei, “Image retrieval using scene graphs,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3668–3678.

[30] K. Xu, J. Ba, R. Kiros, K. Cho, and A. Courville, “Show, attend and tell: Neural image caption generation with visual attention,” in *International Conference on Machine Learning*, 2015, pp. 2048–2057.

[31] B. Alexe, N. Heess, Y. Teh, and V. Ferrari, “Searching for objects driven by context,” in *Advances in Neural Information Processing Systems*, 2012, pp. 1–9.

[32] A. Gonzalez-Garcia, A. Vezhnevets, and V. Ferrari, “An active search strategy for efficient object class detection,” in *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2015, pp. 3022–3031.

[33] D. H. Ballard, “Animate vision,” *Artificial Intelligence*, vol. 48, no. 1, pp. 57–86, 1991.

[34] J. Ba, V. Mnih, and K. Kavukcuoglu, “Multiple object recognition with visual attention,” *arXiv:1412.7755*, 2014.

[35] A. Borji and L. Itti, “State-of-the-art in visual attention modeling,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 185–207, 2013.

[36] V. Mnih, N. Heess, A. Graves, and K. Kavukcuoglu, “Recurrent models of visual attention,” in *Advances in Neural Information Processing Systems (NIPS)*, 2014, pp. 2204–2212.

[37] M. Everingham, L. Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The Pascal visual object classes (VOC) challenge,” *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.

[38] R. Krishna, Y. Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, S. Chen, Y. Kalantidis, L.-J. Li, D. A. Shamma, and Others, “Visual genome: Connecting language and vision using crowdsourced dense image annotations,” *International Journal of Computer Vision*, vol. 123, no. 1, pp. 32–73, 2017.

[39] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, “Microsoft COCO: Common objects in context,” in *European Conference on Computer Vision (ECCV)*, may 2014, pp. 740–755.

[40] “Situate Project Website,” 2015. [Online]. Available: http://www.cs.pdx.edu/$\sim$mm/situate.html

[41] B. Yao, X. Jiang, A. Khosla, A. L. Lin, L. Guibas, and L. Fei-Fei, “Human action recognition by learning bases of action attributes and parts,” in *International Conference on Computer Vision (ICCV)*. IEEE, 2011, pp. 1331–1338.

[42] R. B. Girshick, “FasterRCNN Source Code,” 2015. [Online]. Available: https://github.com/rbgirshick/py-faster-rcnn

[43] P. Krähenbühl and V. Koltun, “Geodesic object proposals,” in *Proceedings of the European Conference on Computer Vision (ECCV 2014)*. Springer, Cham, 2014, pp. 725–739.

[44] D. R. Hofstadter, “Analogy as the core of cognition,” in *The Analogical Mind: Perspectives from Cognitive Science*, D. Gentner, K. J. Holyoak, and B. N. Kokinov, Eds. MIT Press, 2001, pp. 499–538.