Incrementally Biasing Visual Search Using Natural Language Input

Evan Krause[1], Rehj Cantrell[2], Ekaterina Potapova[3], Michael Zillich[3], Matthias Scheutz[1]

Tufts University[1]
Medford, MA USA
{ekrause, mscheutz}@cs.tufts.edu

Indiana University[2]
Bloomington, IN USA
rcantrel@indiana.edu

Vienna University of Technology[3]
Vienna, Austria
{potapova, zillich}@acin.tuwien.ac.at

ABSTRACT

Humans expect interlocutors both human and robot to resolve spoken references to visually-perceivable objects incrementally as the referents are verbally described. For this reason, tight integration of visual search with natural language processing, and real-time operation of both are requirements for natural interactions between humans and robots. In this paper, we present an integrated robotic architecture with novel incremental vision and natural language processing. We demonstrate that incrementally refining attentional focus using linguistic constraints achieves significantly better performance of the vision system compared to non-incremental visual processing.

Categories and Subject Descriptors

I.2.7 [**Artificial Intelligence**]: Natural Language Processing—Discourse, Language parsing and understanding; I.2.10 [**Vision and Scene Understanding**]: 3D/stereo scene analysis; I.4.8 [**Image Processing and Computer Vision**]: Scene Analysis—Color, Depth cues

General Terms

Performance, Human Factors

Keywords

incremental natural language processing, visual search, object detection and recognition

1. INTRODUCTION

Converging evidence in psycholinguistics has demonstrated that humans rapidly and incrementally integrate linguistic and perceptual information in situated contexts, using perceptual information to constrain the syntactic and semantic interpretation of linguistic expressions (e.g., [5]). For example, referential natural language expressions can trigger and guide visual search to determine the intended referent. Conversely, visual analysis of a scene can help disambiguate otherwise ambiguous referential expressions (e.g., referential expressions containing prepositional phrases). Not surprisingly, human speakers have the same expectations of their co-located interlocutors in social interactions (c.f. [4]). Hence, for robots interacting with co-located humans in natural language this means they must respect the human mode of processing. Failing to do so will result in awkward and frustrating interactions at best, but can easily lead to complete interaction break-down.

Respecting human natural language capabilities, however, requires a computational architecture that integrates visual and linguistic information in human-like ways. In this paper, we introduce a tight integration of vision and natural language processing, where linguistic processing incrementally constrains vision. The incremental constraint-based mechanisms are integrated into the DIARC architecture for HRI, which provides an architectural framework for the evaluation of computational mechanisms. We evaluate the effectiveness of incrementality by comparing the operation of two vision processing modes: in the first, a complete description of an object is first generated from natural language input, followed by a single visual search through all existing candidates (i.e., every object in the environment) for the referent; in the second, information gleaned incrementally from natural language input is used to constrain vision's search by progressively narrowing the field of possible candidates, in effect focusing the robot's attention on an increasingly restrictive set of criteria. We show that by so constraining vision, references are resolved significantly faster.

The structure of the paper is as follows. In Section 2, we lay out the overall problem in more detail and review previous work in both natural language processing and vision processing. Then, in Section 3, we introduce our approach to accomplishing the integration of the two types of processing. In Section 4, we describe an experiment that serves to evaluate our approach. We then discuss the results and some limitations of the system as currently implemented in Section 5, closing in Section 6 with a summary of our accomplishments and proposals for future work.

2. MOTIVATION

Imagine a "room cleaning" scenario, where a human instructs a robot to put objects in their proper places in a

Appears in: *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013), Ito, Jonker, Gini, and Shehory (eds.), May, 6–10, 2013, Saint Paul, Minnesota, USA.*

living room. The instructions will likely include object descriptions meant to enable the robot to identify, out of all possible candidate objects, one specific object or set of objects. When instructing a robot, humans will naturally look towards an intended object or point to it, gazing back at the robot to check whether it is attending to the object [22]. If the robot is able to follow the human eye gaze to the target object, both human and robot will establish joint attention which will allow the human instructor to check quickly (and often subconsciously) that the robot understood the request correctly. In addition to looking at the object, humans will typically also expect a robot to verbally acknowledge understanding by saying "OK" or "got it". Feedback is often required even for partial utterances, through eye gaze, verbal acknowledgements, or the immediate initiation of an action such as the robot reaching for a book after hearing "put the red book..." while the utterance is ongoing.

In such an interactive setting, vision and natural language processing can mutually and incrementally constrain each other. For example, visually observing a scene that is being talked about can support understanding of ambiguous or underspecified utterances while they are being processed – "the red book on the floor" will most likely refer to a book visible to the instructor, *not* the one behind her back. Similarly, a syntactically ambiguous sentence like "put the book on the box on the shelf" will become clear as soon as the robot detects a book on the box, thus using visually observed spatial relations to constrain parsing and semantic analysis.

Conversely, incremental processing of a verbal description of a scene can direct visual processing to the relevant elements, e.g., "Put the red [now prioritizing the processing of red image regions] book on [now prioritizing horizontal surfaces on which an object can be placed] the box", or "Take the book on your left [now prioritizing the lower left field of view] ...". In addition, non-linguistic cues such as pointing and gaze direction can be incrementally integrated with partial meanings to steer attention to those elements of the scene relevant to the current discourse situation.

As a robot is carrying out some designated task, it will likely have prior knowledge (e.g., what types of objects and utterances are to be expected in the current scenario and what the roles of objects are) which can effectively constrain visual and language understanding. Situatedness allows the robot to plan actions to get the right perspective and perform active visual search, as well as allowing it to interact with the human (e.g. confirming understanding or asking for more specific object descriptions). Against this background, visual scene and language understanding are tightly coupled, forming the *vision-language loop*. In this loop, language primes vision by modulating attention and visually reconstructed scene elements are fed back as referents for language understanding. These processes are interleaved at a fine temporal granularity to make best use of partial interpretation in both directions.

2.1 Incremental NLP

As described above, human speakers expect co-located listeners to (1) rapidly and incrementally integrate perceptual context (e.g. for reference resolution) (c.f. [4]); and (2) produce backchannel feedback (e.g. eye gaze, verbal acknowledgements such as "okay" and "mhm", and actions like head nodding) that indicates the listener's level of understanding during an ongoing utterance (c.f. [17]). Any robotic NLU system that allows for natural HRI must meet at least these two essential human expectations. In this section, we describe current systems that attempt to handle these requirements to some degree.

While no current robotic NLU systems yet approach the ability to handle natural unrestricted spoken input, several efforts have advanced the state-of-the-art in natural language interactions with artificial entities by tackling different aspects of these challenges. Several robotic systems add genuine NLU components to the robotic architecture (c.f. Michalowski et al.'s GRACE uses a combination of speech and a touch screen [11]; Müller et al.'s semi-autonomous wheelchair responds to coarse route descriptions [14]; Moratz et al. use goal-based or direction-based spoken commands to guide a robot through an environment [13]; Firby's Reactive Action Packages tightly integrate natural language and action execution [6]; and Kruijff et al. [10] are pursing directions in incremental NLU for HRI similar to ours).

However, only a few complete NLU systems operate in real-time. Allen et al. [1] use a manually-designed bottom-up chart parser with preferences and manually-defined weights rather than more standard probabilities. Syntactic analysis is complemented by semantic analysis that returns a logical form as a semantic network. One drawback of this architecture in an HRI setting is its standard pipeline architecture (i.e., syntactic analysis is completed before semantic analysis can begin) which prevents an embodied agent from timely backchanneling. Still more integrated is the system by Schuler et al. [18] which processes phonological, syntactic, and referential semantic information incrementally; however, the system has not been used on a robot.

2.2 Incremental Vision Processing

Visual processing in the presented system serves to identify objects in the scene that are referred to in the dialogue, where we assume that these objects are not known to the system beforehand. So vision has to segment objects relevant to the current discourse from the scene. Several lines of research have addressed the problem of modulated object search and interactive or incremental visual processing.

Unconstrained object segmentation is a notoriously hard and ill-defined problem. Mishra et al. [12] show how a seed point, obtained from user input or attention, together with a log-polar image representation, improves segmentation of 2D and depth images; Johnson-Roberson et al. [8] use a similar technique to segment point clouds in grasping scenarios.

While bottom-up attentional processes are well known, more recent work addresses how top-down cues could bias visual search in a task-dependent manner. Choi et al. [3] train an adaptive resonance theory (ART) network from human labelling to inhibit bottom up saliency for non-relevant image regions. The VOCUS system by Frintrop et al. [7] employs bottom-up (scene-dependent) as well as top-down (target-specific) cues, which are learned from training images, leading to increased search performance. Navalpakkam et al. [15] maximize search speed by incorporating prior statistical knowledge of target and distractor features to modulate the response gains of neurons encoding features.

The concept of incremental visual processing has not received much attention. Typically the aim is simply to make vision methods "as fast as possible". However often not all results are needed immediately or there is a trade-off between speed and accuracy. In one early attempt, Toyama et al. [20] layer so-called "selectors" and "trackers" such that selectors at lower (coarser) levels reduce the set of object candidates for higher levels, with trackers at the top generating output sets of size one. Failure at level i lets the system fall back on layer $i-1$, with a broader search space but smaller accuracy. The system can thus robustly maintain track, adjusting search space and accordingly tracking accuracy to changing conditions. Zillich [23] shows how an incremental approach in the perceptual grouping of edge segments removes the necessity of tuning parameters, which are often difficult to select and tend to lead to brittle systems.

Most related to our work is the work on interaction between vision and language by Bergström et al. [2] and Johnson-Roberson et al. [9] who perform interactive segmentation of 2D images and 3D point clouds based on real-time MRF graph partitioning. Dialogue such as *robot*: "I think there are two objects" *human*: "No there are three objects" or *robot*: "So, should I split the green segment?" *human*: "No, the yellow one!" is used to bias graph partitioning to form the most likely objects. However their work explicitly requires interaction in both ways to refine segmentation, rather than just collecting attentional cues from the human.

While these and related research efforts tackle various aspects of natural language understanding and vision, there is currently no framework that allows for a deep integration of these different algorithms with a complex vision system into a unified integrated robotic architecture for natural HRI.

3. INTEGRATING INCREMENTAL NL AND VISION PROCESSING

The context of situated natural language interactions between humans and robots provides several unique challenges for integrated robotic architectures, in particular, for *visual scene and natural language understanding.* We focus on two:

Challenge 1: Human timing. All visual, natural language and action processing must be performed and completed within human-acceptable timing, ranging from fractions of a second for eye movements and other motor actions, to at most one second for verbal responses.

Challenge 2: Incremental multi-modal constraint integration. All processing must be incremental for the robot to be able to determine the meanings of partial instructions, perform any required perception actions including the establishment of joint attention, and either acknowledge understanding or ask for clarification.

We address these challenges through incremental natural language and vision processing. The former gradually builds a hierarchical semantic representation, requesting a new vision search for each new discourse entity. The latter then allows for the continual refinement of the search by the addition of new filters as additional description is given.

3.1 Incremental NL

The overall incremental natural language system uses a shift-reduce dependency parser (a reimplementation of MaltParser) trained on approximately 2500 sentences comprising the training set (sections 02–21) of the Wall Street Journal (WSJ) corpus. The parser identifies labelled head/argument pairings (e.g., subject/predicate) and identifies, for each word, a manually-created dictionary definition.

These definitions are used to map verbal elements to usable concepts. For example, "red," being a filter that can be used by vision to select certain objects in the real world, is a usable concept. "Scarlet," not being a known filter name, is not. In order for any word to be useful to a perceptual component, the definition must include the perceptual component to which it is useful (in the case of colors, this is vision, as opposed to, say a laser range finder) and whatever the component needs to access the concept (in this case, of course, the filter name "red"). Multiple definitions can point to the same filter, so for example, the word "scarlet" could be given a definition pointing to the red filter.

Similarly, nouns have definitions including what perceptual component can be used to find them. For example, if doors are detected using a laser-range finder rather than with vision, then the vision search is stopped as soon as it is determined that the head of the noun phrase is a non-vision noun. (In this study, the generic noun "object" was used in all cases.) Some nouns have no perceptual attachment and are thus not processed perceptually. Plural and their corresponding singular nouns use the same definitions.

The commonly-used dependency grammar with which WSJ is annotated is not *by itself* suitable for incremental processing of noun phrases (NPs). This is because dependency arcs exist only between the noun head and its dependants (e.g., determiners and adjectives). Thus in the NP "the tall red object", the first three words are not known to be connected in any way — and thus cannot be added to a single search — until "object" is heard. Thus a dependency parser trained on this grammar must be augmented to increase incrementality processing of these phrases.

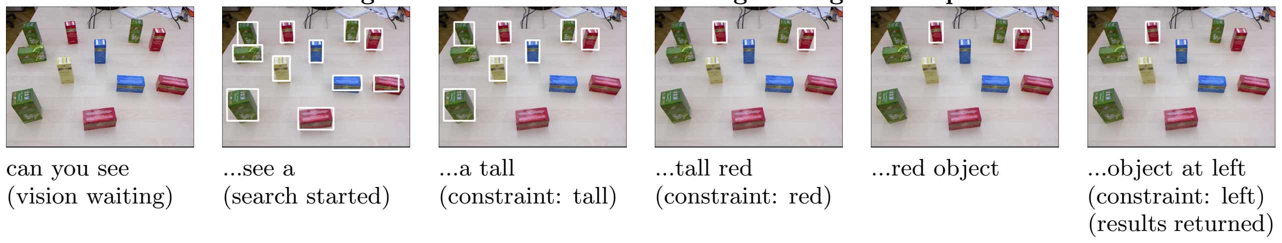
In order to process visual descriptions, our incremental natural language system identifies NPs as beginning with determiners and adjectives[1]. When such a phrase is identified, the system begins a visual search, adding filters as their associated descriptive words are recognized. The system also begins scanning for the end of the phrase. "Descriptive" words are those with a definition that includes a type (e.g., "color") and a filter name (e.g, "red" or "blue"). When the end of the NP is identified (i.e., when a word is found that does not belong to the NP, such as a verb), vision is notified to expect no further constraints, and the results of the search are returned to NLP.

Depending on the number of entities returned and the determiner (or lack of a determiner) that began the noun phrase, one of the following cases occurs: (1) The robot is able to identify one or more objects that meet the description, and it assents, "yes"; (2) the robot is not able to identify any object meeting the description, and it announces, "I could not find any"; or (3) the robot is expecting to find one and only one such object (e.g., "there is *the* [or *one*] blue object"), but it finds multiple such objects, and announces, "I was not able to identify a single referent. Please rephrase the input with a uniquely-identifying set of constraints."

This verification process is used in the case of all types of utterances. Given a situation in which two blue objects are before the robot, if the robot is asked, "Do you see the blue object?" or if it is directed "Pick up the blue object," the robot, being unable to find a single uniquely-identified

[1]NPs beginning with bare nouns, e.g., "objects," do not typically require *immediate* visual processing and thus are not currently handled this way. For example, the command "get objects," rather than triggering an immediate visual search, institutes an open-world quantified goal [] which takes the search out of NLP's purview.

Figure 1: Incremental Processing of Linguistic Input



object meeting the description in either case, will request additional constraints to narrow the reference down.

Given a situation in which no blue objects are before the robot, if the robot is told "there is a blue object," the robot will respond that it cannot find any blue object. It is the determiners that communicate how many objects meeting the description the robot is to expect. The robot is able to distinguish between three types of determiners: existentials (a, an, any, some) which require at least one such object; referentials (the) which require exactly one object; and universals (all, every, each) that allow any number of objects.

Figure 1 shows an example of incremental processing. The determiner is the first sign of a coming noun phrase and the trigger to start a visual search. As vision-relevant words are heard, they are sent as constraints to the vision search.

3.2 Incremental Vision

As described above, vision's goal is to identify the object(s) referred to using natural language. We tackle the problem of segmenting these objects from the overall visual scene by making use of an attention mechanism that relies on cues incrementally obtained from natural language input. The visual input is color images overlaid with 3D point clouds obtained with an RGB-D sensor (a Microsoft Kinect) and organized into a rectangular array (depth image).

The relevant types of vision processors are *saliency operators*, *object detectors* and *object trackers* (see Figure 2). In general, saliency operators detect the amount that a modifier such as a color or a location applies to a particular area of space, while object detectors typically search for specific nouns, such as "faces," "persons," and "objects," which are then tracked by object trackers. An utterance such as "Do you see the blue object?" starts a visual search process composed of a number of saliency operators and one detector and its associated tracker. Several such visual searches can run in parallel. Within a single visual search, visual processors are registered to one another so that the completion of a processing iteration in one processor notifies the other processors that are related to the same search.

3.2.1 Saliency operators

Saliency operators are computationally cheap bottom-up processes that operate on RGB-D images, denoted I. When a saliency operator is created as part of a visual search it is configured using *processing descriptors*, which are essentially the adjectives extracted from the utterance (i.e., which specify what quality is being sought in this specific search). Each saliency operator, S, then outputs a 2D saliency map, M, with values between 0 (not salient) and 1 (maximally salient) overlaid on the 3D point cloud such that $S(I) = M$.

Each saliency operator can be described in terms of its cost per iteration $C(S)$, where a single iteration refers to the processing of an entire image frame.

$$C(S) = c_p * n_p + c_f \tag{1}$$

Here, n_p refers to the number of pixels (or 3D points) that are processed by a given saliency operator, and c_p is the associated cost of processing a single point p. c_f includes additional fixed costs that do not fluctuate with respect to n_p. Given this overall cost structure for saliency operators, we notice that by reducing n_p we can reduce cost and thus decrease CPU consumption and processing time. One way to lower n_p is to enable saliency operators to use the results from completed saliency operators to prune the portion of the image that needs to be processed. Having a vision framework that enables saliency operators to process data in this fashion presents the opportunity to leverage partial search results in a natural and incremental way.

In addition to incremental processing, another critical aspect of the vision framework is that saliency operators are able to run in parallel. This allows saliency operators to interact in four different ways: serial without incremental influence (SN), serial with incremental influence (SI), parallel without incremental influence (PN), and parallel with incremental influence (PI) (see Figure 3). Take, for example, the search for a "tall red object," which requires both height (S_h) and color (S_c) saliency operators. In SN, S_h processes the entire image, assigning a saliency value to every pixel in the resulting saliency map M_h, followed by S_c also processing the entire image. In SI, the second saliency operator S_c uses M_h to only search for "red" in the region(s) of the image corresponding to "tall."

By contrast, in PN, S_h and S_c process the image simultaneously in independent threads. PI is similar, except that saliency operators are also able to notify and interrupt other operators when they have completed an iteration. If, for ex-

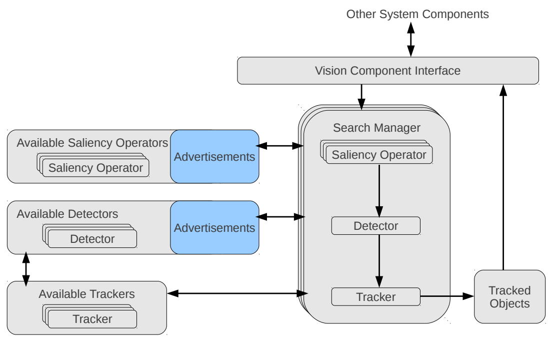


Figure 2: A high-level view of the vision framework.

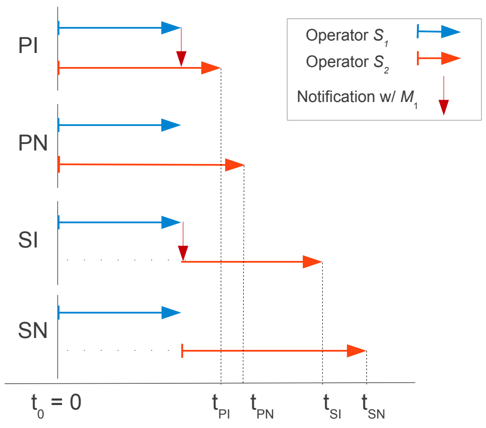


Figure 3: Four saliency operator configurations and their relative times to completion

ample, S_h and S_c are running in parallel and S_c is a much faster processor, it can notify S_h of its completion potentially speeding up the remaining portion of the S_h iteration.

For a given visual search task, we would generally expect SN to have the longest time to completion, followed by SI, then PN, with PI being the most efficient, as shown in Figure 3. It is worth noting, however, that not every saliency operator will lend itself to performance improvements via incremental processing. If an operator has a very cheap c_p, there's little to be gained from incremental processing. In the case of PI, additional constraints must be met in order to see a performance gain over the non-incremental case. If, for example, two operators start at the same time and have a similar iteration time, neither can benefit from the other because both will have completed their iteration before information from the other becomes available. Thus, in addition to meeting constraints on c_p, operators will also need to have significantly different processing times or start in a staggered fashion to realize performance gains in PI.

Note that each saliency operator in a visual search need not be very distinctive, certainly not enough to in itself suffice for segmentation. Furthermore, some might be ambiguous (e.g., "short" could be meant as the opposite of "tall" or refer to the small length of an elongated object). So we do not expect each of these operators to output very precise information. All these operators need to do in common is to prioritize salient image regions (and thus corresponding parts of the point cloud) in order to render the following segmentation step computationally more tractable.

3.2.2 *Object detection*

Objects are detected by segmenting the 3D point cloud. We make the simplifying assumption often used in robotics scenarios [8, 21] that objects are located on a dominant supporting plane. Segmentation then amounts to detecting the supporting plane, subtracting it from the point cloud, and clustering the remaining points into object candidates. Clustering is based on the Euclidian clustering method provided by the Point Cloud Library (PCL) [16] and is computationally the most expensive step.

To determine which object cluster(s) in the scene correspond to the utterance, the detector validates each candidate cluster using the saliency maps. It operates in one of two configurations: without or with attention. The first approach, without attention, blindly iterates through the object clusters, checking each cluster against the saliency maps, stopping the detector iteration only after the whole point cloud has been processed. The second approach sorts 3D points in order of decreasing saliency and uses a modification of the PCL Euclidian clustering method to start with the most salient point, greedily collect neighbouring points and output the first most salient cluster, then repeat. Thus the most salient objects pop out first and are immediately available as referents for language, while less salient objects follow later. This has the additional advantage of (potentially) terminating the search before all clusters are checked. This is simply done by stopping the cluster validation when the sorted saliency map has no saliency values remaining.

In order to bind detected objects as visual referents a final decision has to be made whether an object does indeed meet the description, e.g., is it truly blue and is it truly tall. This decision is based on thresholds, performed once objects have been segmented. While the decision could have been decided based on the saliency maps themselves, doing so was avoided because the output of saliency operators cannot be considered very precise. Furthermore, meaningful thresholds are difficult to define and will change from scene to scene.

3.2.3 *Object tracking*

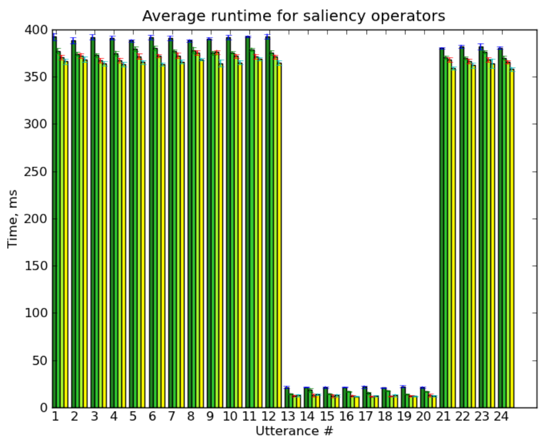
Once a detector has successfully segmented objects from a scene, a tracker is tasked with consuming the resulting objects and tracking them from frame to frame. Object tracking is performed by associating previously found objects with new ones based on spatial consistency. Two objects are considered to be equal if they overlap by more than 50%, otherwise a new object is added to the tracker.

These processors work in tandem with each other and share information. A visual search for a "tall red object," for instance, might consist of an "object" detector using the results from a "red" saliency operator and a "tall" saliency operator. However, these implementation details must interact transparently with outside components such as natural language without burdening them with vision-internal details. Such transparent interaction is provided by the interface described in the next subsection.

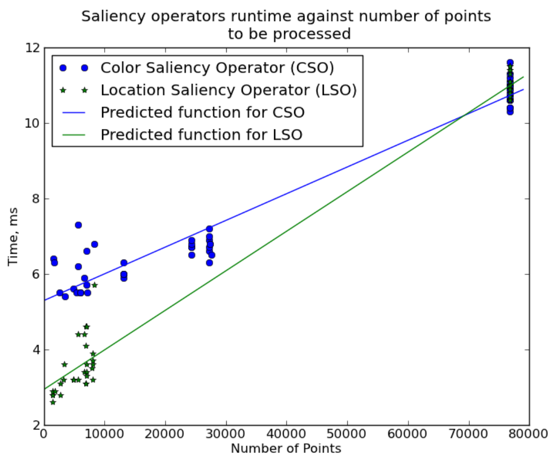
3.3 The Interface between Vision and NL

In order for a robotic system to perform naturally in the context of human-robot interactions, a robot vision system needs to be able to quickly respond to incremental cues from natural language in order to dynamically instantiate and modify visual searches. To accomplish this, a vision system needs to expose an interface capable of naturally handling requests from natural language components, thereby freeing language components from requiring an intimate knowledge of visual components and their capabilities. A common currency must exist between language and vision components to enable this timely and natural interaction. Additionally, the vision framework must be able to rapidly convert requests (possibly incomplete) from natural language into meaningful visual searches in a robust and dynamic way.

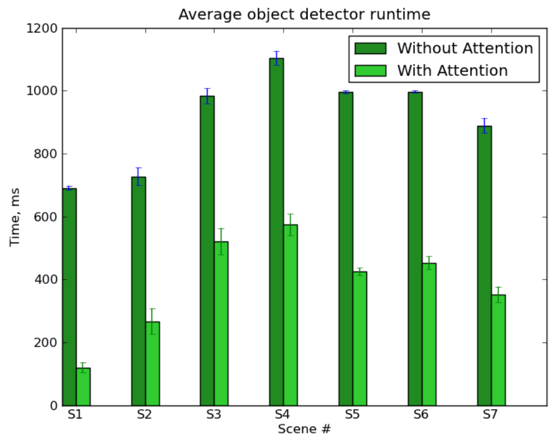
The interface between natural language and vision is handled by *search managers*, the highest level mechanism responsible for dynamically building searches from natural language cues, which are used to shield outside components



(a) Average runtime and confidence bars for all saliency operators to complete for a given utterance. 24 utterances are presented for a single scene, where each utterance is ran in all four configurations (left to right: SN, SI, PN, PI). 13-20 are significantly shorter because they do not involve height saliency, which is computationally more expensive.



(b) Runtime of color and location saliency operators plotted against number of processed pixels/points.



(c) Average object detector runtime across scenes.

Figure 4: Experimental results.

from internal implementation details. When a new visual search is triggered by an outside component via a call to *createNewType*, a new search manager is instantiated, and a unique search ID returned to the caller so that future visual constraint requests can be associated with that particular search. *addDescriptor* is then used to populate the search with the appropriate combination of saliency, detector, and tracker without the outside component being required to know any of the details of the different processor types.

Because each processor has a unique capability depending on its underlying implementation, processors are responsible for advertising their capabilities to the search manager in order to allow it to populate the search with the appropriate vision processors[2]. For example, a processor capable of generating saliency maps for various color values might advertise "red," "green," and "blue." These advertisements are specified at runtime via a series of xml configuration files. In keeping with the responsibilities of different types of processors as described in the previous subsection, detector advertisements are generally nouns, as opposed to the description-based advertisements of saliency operators. In this way the distinction between saliency operators and detectors is hidden within the vision framework, and outside callers are not required to have knowledge about their differences. Search managers automatically route incoming predicates to the most appropriate vision component.

Once objects have been detected and reach the tracking stage, outside components (e.g., NL) can query vision to retrieve results about the visual search (e.g., by calling *getTokensByTypeId*). Once a visual search is no longer needed, a request to vision to *stopAndRemoveType* can be made, which stops all vision components related to that particular search, and returns the resources back to the system.

To summarize, as a search manager receives incremental constraints, the manager maps the incoming predicate to the most appropriate vision component (i.e., detector or saliency operator), instantiates it, and starts its processing loop. An arbitrary number of constraints can be incrementally added to a search. A fully functional visual search is composed of a detector, tracker, and zero or more saliency operators. Clients of vision are freed from knowing details of the underlying vision framework, and need only provide a search ID and predicate constraints to build visual searches and query for results.

4. EXPERIMENTAL EVALUATION

We evaluated the effectiveness of language-modulated attention by measuring the time needed to identify a specific discourse referent. In particular, we separately measured the processing time of saliency operators and detectors to fully evaluate the various vision configurations. We constructed 7 scenes, each composed of eleven objects, which were subsequently referred to in utterances such as "Can you see a tall red object on the left?" For the constructed scenes, only the following three saliency operators were required to uniquely identify each target object.

Color saliency. The color saliency operator maintains a list of commonly used color words ("blue", "red", "black")

[2]This does not apply to those processors that are only used internally by other vision processors, e.g., trackers, which are paired with object detectors and thus do not advertise their capabilities to the system.

associated with points in color space. These associations are currently hand-coded but could also be learned as in [19]. Distances of pixel colors to the salient color selected by the processing descriptor are mapped to saliency values in $[0, 1]$. Note that several colors can be salient at the same time ("the red or blue object"), in which case the minimum distance to a salient color is used.

Location saliency. Another commonly used property when talking about a scene containing several objects is relative location, as in "Pick up the object on the left". The location saliency operator maps location in the image to saliency and can be configured for "center", "left", "right", "top" or " bottom". Saliency decreases in the form of a Gaussian from the selected image border or image center.

Height saliency. While operating on the raw input data prior to segmentation does not allow specification of object shape properties (as we have not segmented any objects yet) object height above ground (i.e., the supporting surface) is a simple cue to support properties such as "tall" and "short". Height from ground to the highest point above ground is mapped to $[0, 1]$ for "tall" and $[1, 0]$ for "short" respectively.

Each scene (object configuration) was paired with a set of eleven utterances, each uniquely identifying a different target object within the scene. Each scene/utterance run was repeated ten times as processing time was measured. Four vision configurations were used: (SN,PN) without detector attention, and (SI,PI) with detector attention. In order to formally evaluate the various vision configurations, the experimental results were collected without the natural language components, instead using a pre-processed version of the utterances. This allowed the precise timing of vision processing to be analyzed without being influenced by the timing constraints imposed by language processing.

The results are presented in Figures 4(a) to 4(c). Figure 4(a) shows the average time and confidence for all saliency operators to complete for a given search utterance. The times presented are for eight different utterances[3], and all permutations (e.g.,"short blue" and "blue short"), for a single scene. Each utterance/search was performed ten times for each of the four configurations. Our predictions about the relative runtimes from Section 3.2.1 are met. For each utterance, SN has the longest runtime, followed by SI. Additionally, the parallel cases have shorter runtimes than the serial cases. Notice, however, that t_{PI} is not always less than t_{PN}. This is attributable to the fact that the saliency operators used in these experiments do not meet the criteria laid out in Section 3.2.1, namely that operators need to end at staggered times and also have a high enough c_p. Additionally, because the runtime improvements across configurations greatly depends on the particular vision process, more expensive processes can potentially benefit a great deal.

Figure 4(b) shows the runtime of the color and location operators as a function of n_p. The slope of the lines corresponds to c_p, showing that using incrementally available results to prune the search space (decreasing n_p) is an effective way to reduce processing times of saliency operators.

Figure 4(c) shows the average time and confidence of the object detector for each scene, averaged over all target objects. The times for both detector configurations (i.e., with and without attention) are compared, and it is clear that using attention significantly speeds up detection time. Speedups vary for different objects or positions in the image (e.g. large objects near the top are typically found first when processing the image pixel row for pixel row, so speedup for those objects is smaller), but with attention the target object is typically found first, while without attention it is on average found after checking half of the objects.

[3]Searches involving only a single saliency operator, and searches giving occasional incorrect search results are not shown.

Holding saliency operators constant, each dimension was tested individually for significance. Holding ordering (serial or parallel) constant and varying incrementality, incrementality showed main effects in both serial and parallel conditions ($F_{Par}(1, 238) = 112.56, p < 0.001; F_{Ser}(1, 238) = 680.97, p < 0.001$). Holding incrementality constant and varying ordering, ordering showed main effects in both incremental and nonincremental conditions ($F_{Inc}(1, 238) = 405.52, p < 0.001; F_{Non} = 1130.1, p < 0.001$). Hence both dimensions were instrumental in yielding gains.

These results clearly demonstrate that the attentional cues obtained incrementally from dialogue efficiently steer visual processing to the relevant parts of the scene, resulting in significantly reduced runtimes for detecting the target object.

5. DISCUSSION

While the experimental results focus on the vision component and its ability to operate under various configurations at varying degrees of efficiency, we have also validated our approach using full utterances in a combined Vision and NLP configuration. A full discussion of these results, along with how these two components integrate within a larger robotic architecture is beyond the scope of this paper. We instead focus on the challenges and shortcomings encountered with these two components.

One challenge (shared by most if not all parsing systems) is the attachment of prepositional phrases. Given a phrase such as "Put the tall red object on the table in the other room," it is unclear whether the table is in this room or the other room. In the former case, "on the table" should be included in the description of the red object; in the latter case it should be included in the instruction of where to place the object. Its importance and a possible solution are offered by the fact that people often produce as terse a description as possible (i.e., they say only what is necessary to uniquely identify the object in question): if no object is found, the system should have the ability to experimentally discard constraints one by one until a match is found.

On the vision side, the evaluation assumed that each configuration was precisely achieved. In practice, the exact configurations and resulting search time will depend heavily on the timing of the search utterance (specifically the time between visual descriptors), and factors such as whether or not the target item is in the field of view while descriptors are being parsed. The distinction between serial and parallel processing will likely be far less rigid. The configuration of saliency operators and their interactions will not be decided by a predetermined vision configuration, but will instead be dynamically determined by the nature of the human-robot interaction. For this reason, it is critical for the vision framework to be able to robustly and dynamically handle all these configurations and the configurations that fall between them.

It is worth noting that in the experiments presented here, the visual processing required was fairly simple. Once we eliminate initial simplifying assumptions and add increasingly complex vision processes, non-attentional approaches are bound to hit performance bottlenecks. Biological vision systems have developed attentional mechanisms to be able to quickly react to the relevant parts of the visual scene. Accordingly, the focus in our work lies in developing principled attentional mechanisms for the case of human robot interaction to support visual processing at time frames compatible with human language, rather than in optimising specific vision methods for certain scenarios.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we argued for integrated incremental vision and natural language processing in order for robots to meet the requirements posed by natural interactions with humans. We demonstrated experimentally that constraining vision with incrementally-acquired natural language descriptions significantly speeds vision processing, and thus also reference resolution. In addition, future work will explore the reverse direction, using visually-acquired information to constrain natural language interpretation.

Also to be addressed is the threshold value used to decide whether a detected object meets the description. Future work will employ probabilistic models of object properties (such as the incrementally learned KDE based representations of Skocaj et al. [19]). An extension of the NLU system will fuse existing confidence measures (e.g., in a parse tree) with such probabilistic measures from the vision system.

7. ACKNOWLEDGMENTS

This work was in part funded by ONR grant N00014-11-1-0493 and the Austrian Science Fund (FWF) under project TRP 139-N23 InSitu.

8. REFERENCES

[1] J. Allen, M. Dzikovska, M. Manshadi, and M. Swift. Deep linguistic processing for spoken dialogue systems. In *Proceedings of the ACL Workshop on Deep Linguistic Processing*, Prague, Czech Republic, 2007.

[2] N. Bergstrom, M. Bjorkman, and D. Kragic. Generating Object Hypotheses in Natural Scenes through Human-Robot Interaction. In *IROS*, pages 827–833, 2011.

[3] S.-B. Choi, S.-W. Ban, M. Lee, J.-K. Shin, D.-W. Seo, and H.-S. Yang. Biologically motivated trainable selective attention model using adaptive resonance theory network. In A. Ijspeert, M. Murata, and N. Wakamiya, editors, *Biologically inspired approaches to advanced information technology*, pages 456–471. Springer Berlin / Heidelberg, 2004.

[4] H. Clark and C. Marshall. Definite reference and mutual knowledge. In A. K. Joshi, B. L. Webber, and I. A. Sag, editors, *Elements of discourse understanding*, pages 10–63. Cambridge University Press, Cambridge, 1981.

[5] K. M. Eberhard, M. J. Spivey-Knowlton, J. C. Sedivy, and M. K. Tanenhaus. Eye movements as a window into real-time spoken language comprehension in natural contexts. *Journal of Psycholinguistic Research*, 24:409–436, 1995.

[6] R. J. Firby. *Adaptive Execution in Complex Dynamic Worlds*. PhD thesis, Yale University, 1989.

[7] S. Frintrop, G. Backer, and E. Rome. Selecting what is important: Training visual attention. In *Proc. of the 28th Annual German Conference on AI (KI'05)*, 2005.

[8] M. Johnson-Roberson, J. Bohg, M. Bjorkman, and D. Kragic. Attention based active 3D point cloud segmentation. In *IROS*, 2010.

[9] M. Johnson-Roberson, J. Bohg, G. Skantze, J. Gustavson, R. Carlsson, and D. Kragic. Enhanced Visual Scene Understanding through Human-Robot Dialog. In *IROS*, pages 3342–3348, 2011.

[10] P. Lison and G.-J. M. Kruijff. Efficient parsing of spoken inputs for human-robot interaction. In *RO-MAN*, 2009.

[11] M. P. Michalowski, S. Sabanovic, C. DiSalvo, D. B. Font, L. Hiatt, N. Melchoir, and R. Simmons. Socially distributed perception: GRACE plays social tag at AAAI 2005. *Autonomous Robots*, 22(4):385–397, 2007.

[12] A. Mishra and Y. Aloimonos. Active Segmentation for Robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009.

[13] R. Moratz, K. Fischer, and T. Tenbrink. Cognitive modeling of spatial refernce for human-robot interaction. *International Journal on Artificial Intelligence Tools*, 10(4):589–611, 2001.

[14] R. Müller, T. Rofer, A. Landkenau, A. Musto, K. Stein, and A. Eisenkolb. Coarse qualitative description in robot navigation. In C. Freksa, W. Braner, C. Habel, and K. Wender, editors, *Spatial Cognition II*, pages 265–276. Spinger-Verlag, 1998.

[15] V. Navalpakkam and L. Itti. A theory of optimal feature selection during visual search. In *Proc. Computational and Systems Neuroscience*, Mar 2006.

[16] R. B. Rusu and S. Cousins. 3D is here: Point Cloud Library (PCL). In *ICRA*, 2011.

[17] D. Schiffrin. *Discourse Markers*. Cambridge University Press, 1988.

[18] W. Schuler, S. Wu, and L. Schwartz. A framework for fast incremental interpretation during speech decoding. *Computational Linguistics*, 35(3), 2009.

[19] D. Skocaj, M. Janicek, M. Kristan, G.-J. M. Kruijff, A. Leonardis, P. Lison, A. Vrecko, and M. Zillich. A basic cognitive system for interactive continuous learning of visual concepts. In *ICRA 2010 Workshop ICAIR - Interactive Communication for Autonomous Intelligent Robots*, pages 30–36, 2010.

[20] K. Toyama and G. Hager. Incremental focus of attention for robust visual tracking. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition*, pages 189–195, 1996.

[21] W. Wohlkinger and M. Vincze. Shape-Based Depth Image to 3D Model Matching and Classification with Inter-View Similarity. In *IROS*, pages 4865–4870, 2011.

[22] C. Yu, M. Scheutz, and P. Schermerhorn. Investigating multimodal real-time patterns of joint attention in an hri word learning task. In *HRI*, pages 309–316, 2010.

[23] M. Zillich. Incremental Indexing for Parameter-Free Perceptual Grouping. In *Proc. of the 31st Workshop of the Austrian Association for Pattern Recognition (OAGM/AAPR)*, pages 25–32, 2007.