Coupling Robot Perception and Online Simulation for Grounding Conversational Semantics

Deb Roy, Kai-Yuh Hsiao, and Nikolaos Mavridis

***Abstract*— How can we build robots that engage in fluid spoken conversations with people, moving beyond canned responses and towards actual understanding? Many difficult questions arise regarding the nature of word meanings, and how those meanings are grounded in the world of the robot. We introduce an architecture that provides the basis for grounding word meanings in terms of robot perception, action, and memory. The robot's perceptual system drives an online simulator that maintains a virtual version of the physical environment in synchronization with the robot's noisy and changing perceptual input. The simulator serves as a "mental model" that enables object permanence and virtual shifts of perspective. This architecture provides a rich set of data structures and procedures that serve as a basis set for grounding lexical semantics, a step towards situated, conversational robots.**

***Index Terms*— Robots, Natural language interfaces, Knowledge representation, Active vision, Simulation.**

I. Introduction

LANGUAGE enables people to talk about the world. Through language, we are able to refer to the past and future, and to describe things as they are or how we imagine them. For a robot to use language in human-like ways, it must ground the meaning of words in its world as mediated by perception, action, and memory. Many words that refer to things in the world can be grounded through sensory-motor associations. For instance, the meaning of *ball* includes perceptual associations that encode how balls look and predictive models of how balls behave. The representation of *touch* includes procedural associations that encode how to perform the action, as well as perceptual encodings to recognize the action in others. Words thus serve as labels for perceptual or action concepts that are anchored in sensory-motor representations. When a word is uttered, the underlying concept is communicated to the degree that the speaker and listener maintain similar associations. This basic approach underlies most work to date in building machines that ground language [1]–[8].

Not all words, however, can be grounded in sensory-motor representations. In even the simplest conversations about everyday objects, events, and relations, problems arise. Consider a person and a robot sitting across a table from each other, engaged in coordinated activity involving manipulation of objects. After some interaction, the person says to the robot:

Touch the heavy blue thing that was on my left.

To understand and act on this command, the robot must bind words of this utterance to a range of representations:

touch	can be grounded in a visually-guided motor program that enables the robot to move towards and touch objects. This is an example of a procedural association that depends on perception to guide action.
heavy	specifies a property of objects which involves *affordances* [9]. A light object affords manipulation whereas a sufficiently heavy one does not. To represent affordances, both procedural and perceptual representations must be combined.
blue	specifies a visual property, an example of perceptual grounding.
thing	must be grounded in terms of both perception and affordances (one can see an object, and expect to reach out and touch it).
was	triggers a shift of perspective in time. Words and inflections that mark tense must cue the robot to "look" back in time to successfully ground the referent of the utterance.
my	triggers a shift of perspective in space. As opposed to *your left* or simply *left* (which would be ambiguous), *my* tells the listener to look at the world from the speaker's point of view.
left	can be grounded in visual features that compute linguistically-salient spatial relations between objects within an appropriate frame of reference.

We have developed an architecture in which a robotic manipulator is coupled with a physical simulator. By virtue of the robot's sensory, motor control, and simulation processes, a set of representations are obtained for grounding each of the kinds of words listed above[1].

The robot, called Ripley (Figure 1), is driven by compliant actuators which enable it to manipulate small objects. Ripley has cameras, touch, and various other sensors on its "head". Force and position sensors in each actuated joint provide a sense of proprioception. Ripley's visual and proprioceptive systems drive a physical simulator that keeps a constructed version of the world (including a representation of itself) in synchronization with Ripley's noisy perceptual input. An object permanence module determines when to instantiate and

[1]We acknowledge that the words in this example, like most words, have numerous additional connotations that are not captured by the representations that we have suggested. For example, words such as *touch*, *heavy* and *blue* can be used metaphorically to refer to emotional actions and states. *Things* are not always physical perceivable objects, *my* usually indicates possession, and so forth. Barwise and Perry use the phrase "efficiency of language" to highlight the situation-dependent reusability of words and utterances [10]. Given the utterance and context that we described, the groundings listed above are sufficient. Other senses of words may be metaphoric extensions of these embodied representations [11].

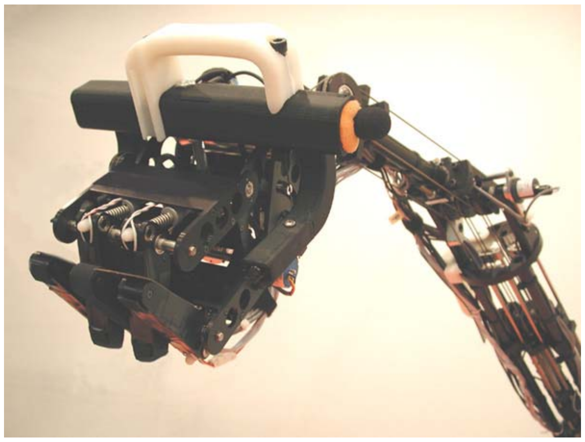

Fig. 1. Ripley has 7 degrees of freedom powered by series elastic actuators enabling it to manipulate objects in a 3 foot radius workspace. The perceptual system includes stereo vision, audition, touch and proprioceptive sensors, and a sense of gravity. The camera and microphones are placed on Ripley's 'head'. Ripley is shown here looking up from the table.

destroy objects in the model based on perceptual evidence. Once instantiated, perception can continue to influence the properties of an object in the model, but knowledge of physical world dynamics is built into the simulator and counteracts 'unreasonable' percepts. The simulator maintains a consistent and stable interpretation of sensory experience over time.

Language can be grounded in terms of associations with elements of this perceptually driven world model. Although the world model directly reflects reality, the state of the model is the result of an interpretation process that compiles perceptual input into a stable registration of the environment. As opposed to direct perception, the world model affords the ability to assume arbitrary points of view through the use of synthetic vision which operates within the simulator, enabling a limited form of "out of body experience". This ability is essential to successfully differentiate the semantics of *my left* versus *your left*. Non-linguistic cues such as the visual location of the speaker can be integrated with linguistic input to trigger context-appropriate perspective shifts. Shifts of perspective in time and space may be thought of as *semantic modulation functions*. Although the meaning of *left* in one sense remains constant across usages, words like *my* and *your* modulate their meaning by shifting frames of reference. We believe that fluid use of language requires constant modulation of grounded meanings.

In this paper, we describe the robot and simulator, and mechanisms for real-time coupling. We then discuss mechanisms designed for the purpose of grounding the semantics of situated, natural spoken conversation. Although a language understanding system has not yet been constructed, we conclude by sketching how the semantics of each word within the utterance discussed above can be grounded in data structures and processes provided by this architecture. This work provides a foundation for our long term goal of developing robots and other machines that use language in human-like ways by leveraging deep, grounded representations of meaning that hook into the world through robot perception, action, and higher layers of cognitive processes.

Conversational robots may be of use in deep sea and space exploration, remote handling of hazardous materials, and assistive health care. Beyond conversational robots, the underlying ideas of how to endow machines with a deeper sense of word meaning may have numerous applications ranging from question answering systems to language translation.

II. Background

Although robots may be connected to speech recognizers, parsers, and speech synthesizers in various simple ways, the results are typically limited to canned behavior. Attempts to endow robots with linguistic capabilities introduces deep theoretical issues that touch on virtually all aspects of artificial intelligence (and cognitive science) including perception, action, memory, and planning. Indeed, the problem of combining robots and language reminds us of the impossibility of treating language in isolation from numerous non-linguistic processes.

The term *grounding* is often used to refer to the process of anchoring the meaning of words and utterances in terms of non-linguistic representations that language users possess through a combination of evolutionary and lifetime learning. The issue of language grounding, in various forms and carrying numerous labels, has been a central issue in the philosophy of language and mind since antiquity. In contrast, most work on natural language processing (NLP) by machine tends to focus on structural / relational representations of word meaning, defining words through typed relations to other words (cf. [12]), much in the spirit of conventional dictionaries. Jurafsky and Martin, authors of a recent text on NLP, point out the fundamental limitation of such approaches:

> This inherent circularity [of defining words in terms of other words] is evidence that dictionary entries are often not definitions at all, but rather descriptions of lexemes in terms of other lexemes. Such entries are useful when the user of the dictionary has sufficient grasp of these other terms to make the entry in question sensible. As is obvious with lexemes like *red* and *right*, this approach will fail without some ultimate grounding in the external world [13] (p. 591).

For robots to understand and use language, we must turn to methods beyond the standard techniques of NLP so that these machines can connect words and utterances to actions and percepts in the physical world. One approach would be to connect words to perceptual classifiers so that the appearance of an object, event, or relation in the environment can instantiate a corresponding word in the robot [8], [14], [15]. Crangle and Suppes [16] also take this approach, and furthermore present a detailed approach to integrating syntactic processing of commands with robot control programs.

Detailed models have been suggested for sensory-motor representations underlying color [6], and spatial relations [3], [17]. Models for grounding verbs include grounding verb meanings in the perception of actions [5], and in terms of motor control programs [1], [2]. Although object shape is important when connecting language to the world, it remains a challenging problem for computational models of language grounding. In previous work, we found histograms of local geometric features sufficient for grounding names of basic objects (dogs, shoes, cars, etc.) [18]. Landau and Jackendoff provide a detailed analysis of additional visual shape features, such as axes of orientation, that also play a role in language [19].

Clark observed that people speak and gesture to coordinate *joint actions* [20]. Speakers and listeners use various aspects of their physical environment to encode and decode utterance meanings. Communication partners are aware of each other's gestures and foci of attention and they integrate these sources of information in conversation. Even young infants leverage situational cues and knowledge of joint attention to facilitate language acquisition [21]. Recent work on social robots have explored mechanisms that provide visual awareness of human partner's gaze and other facial cues relevant for interaction [22]–[24]. Although our current architecture does not include representations of social cues, Clark's work highlights the need for eventually incorporating such cues into a model of language grounding (for a computational approach along these lines, see [25]).

To understand language, it is often necessary for the listener to assume the speaker's point of view. Points of view can involve seeing something from a different physical vantage point, as well as less literal perspective shifts in order to see things from different social or cultural perspectives. Occasionally, one communication partner carries more of the burden of shifts of view than the other. When speaking to a young child, an adult is more likely to take the view of the child than vice versa. However, for all but the simplest interactions, the child must also recognize the divergence of viewpoints. If we are to build robots that can interact naturally with people, they will need similar perceptual awareness of the environment and communication partners, and the ability to change points of view.

III. Ripley: An Interactive Robot

Ripley has been designed to explore questions of grounded language and interactive language acquisition. The robot has a range of motions that enable it to examine and manipulate objects on a tabletop workspace. Ripley can also look up and make eye contact with a human partner. Three primary considerations have driven the design of the robot. We were interested in studying:

- The effects of changes of visual perspective and their effects on language and conversation.
- Sensory-motor grounding of manipulation verbs (e.g., *touch, lift, push, take,* etc.).
- Human-directed training of motion. For example, to teach Ripley the sensory-motor meaning of *touch*, exemplars of the word can be demonstrated by a human trainer while labeling the behavior.

To address the issue of visual perspective, Ripley has video cameras placed on its head so that all motions of the body lead to changes in view points. This design decision leads to challenges in maintaining stable perspectives in a scene, but reflects the type of corrections that people must also constantly perform that shape how language is used. To support acquisition of verbs, Ripley has a gripper that can grasp objects to enable manipulation. As a result, the most natural class of verbs that Ripley will learn involve manual actions such as touching, lifting, pushing, and giving. To address human directed training, Ripley is actuated with compliant joints, and has training handles (Figure 1).

A. Mechanical Structure and Actuation

Ripley has seven degrees of freedom (DOF's). At the base of the robot, two revolute joints are arranged along perpendicular axes, giving the robot 180 degrees of motion about the two horizontal axes at its base and serving as a waist for the torso (or shoulder if thought of as an arm). Halfway between the base and the robot's head, a single joint with an 180-degree range of motion gives the robot the articulation necessary to touch any point on a work surface in front of it. Ripley's head is connected to the body through three orthogonal DOF's that enable the gripper to orient freely within a 180-degree range about two axes at the end of the arm and to rotate 540 degrees about the axis that runs along the arm. Finally, the gripper itself opens and closes, providing the seventh DOF.

Each DOF other than the gripper is actuated by series-elastic actuators [26] in which all forces from electric motors are transferred through torsion springs. Each motor drives a threaded shaft. A sliding nut fit on the shaft connects to a spring. The other end of the spring is connected to a cable assembly that provides actuation to the robot. This arrangement enables the force applied by the motors to be controlled directly, in contrast to motors which are controlled by speed. Compression sensors are placed on each spring and used to provide force feedback. The use of series-elastic actuators provides precise sensation of the amount of force applied at each DOF, and leads to compliant motions. The springs in each DOF also act as natural shock absorbers that dampen vibrations.

B. Motion Control

Motion control in Ripley is inspired by studies of motor force fields in frogs [27]. In essence, frogs use a sequence of control points to control motions. Local disturbances to motion are compensated by internal force fields which exhibit stronger pull towards control points as a function of distance between actual and desired location. As a rough approximation to this method, a position-derivative control loop is used to track target points that are sequenced to move smoothly from the starting point of a motion gesture to the end point. Natural motion trajectories are learned from human teachers through manual demonstrations.

The robot's motion control is organized in a layered architecture. The lowest level of control is implemented in hardware and consists of a continuous control loop between motor amplifiers and force sensors of each DOF. At the next level of control, a dedicated microcontroller running a real-time operating system executes a position-derivative (PD) control loop with a 5 millisecond cycle time. The microcontroller controls a set of digital-to-analog converters

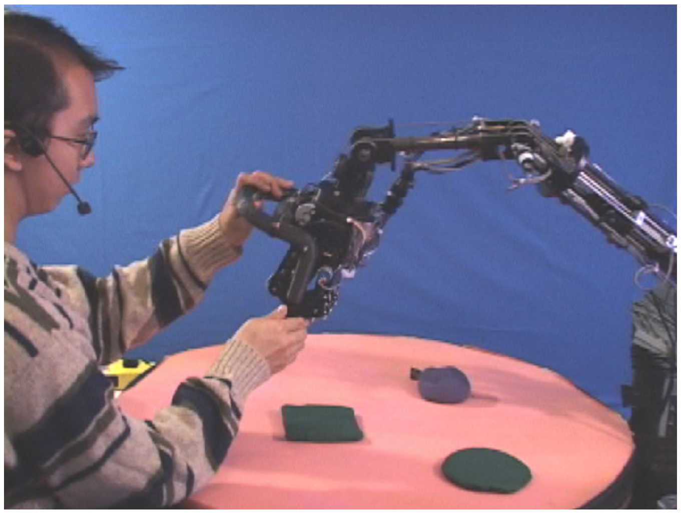

Fig. 2. Ripley's motions can be trained through direct manipulation. The motor controllers are placed in a "anti-gravity" mode so that a human trainer can effortlessly guide Ripley through natural motions. Motion trajectories recorded during training are processed by interpolation algorithms that are combined with visual guidance to generate novel motion trajectories.

(DAC's) that set target force levels for each motor amplifier. The microcontroller also accepts target positions from a master controller (host computer) and translates these targets into force commands via the PD control loop. The resulting force commands are sent downstream to the motor amplifier control loop. The same force commands are also sent upstream to the master controller, serving as dynamic proprioceptive force information.

The master controller is responsible for providing target point trajectories to the microcontroller. A set of seven target positions, one for each DOF, are sent to the microcontroller every 20 milliseconds. We refer to a stream of position target points as a *motion trajectory*. Motion trajectories can be created by manual demonstration. A human trainer simply grasps Ripley's training handles and demonstrates a motion much like the way an adult might show a child how to perform an action by guiding his/her hand along an ideal trajectory (Figure 2). This approach provides a simple and intuitive means for specifying fluid, natural motions which takes into account the natural affordances of the robot's mechanical design, as well as introducing "puppeteer" instincts of human trainers to capture the aesthetics of smooth, natural motions.

To support training, we have implemented a special mode of motor control referred to as *anti-gravity mode*. This allows Ripley to remain active and hold its own weight, while being responsive to light touch from the human trainer. This eases the physical force that the trainer must exert to move Ripley, resulting in more natural movement trajectories.

To achieve anti-gravity motor control, joint position information is used to estimate the force of gravity on each DOF of the robot. An approximate model of the mass distribution of the robot is used to estimate the effects of gravity. Since the robot's actuators compensate with an equal force on each DOF, the robot can be made to stay perfectly still in any position, effectively canceling the effect of gravity. However, in such a state, the robot still responds perfectly to other external forces, resulting in remarkable pliability. By leading the seemingly lightweight robot through gestures, it is possible for human trainers to easily generate motion trajectories.

Motion trajectories are created by recording the outputs of the position sensors during training. These control points can then later be sent to the microcontroller for motion playback. Motion trajectories can be interrupted and smoothly revised to follow new trajectories as determined by higher level control. We have also implemented interpolative algorithms that blend trajectories to produce new motions beyond the training set. In Section D below, we describe how motion trajectory demonstrations are combined to teach Ripley how to reach for objects in arbitrary positions.

C. Sensory System and Visual Processing

Ripley's perceptual system is based on several kinds of sensors. Two color video cameras, a two-axis tilt accelerometer (for sensing gravity), and two microphones are mounted on the head. Force sensitive resistors provide a sense of touch on the inside and outside surfaces of the gripper. In the work reported here, we make use of the visual, touch, and force sensors. The remaining sensors will be used in future applications.

Some of the most important sensors are embedded in the actuators which are force-controlled, meaning that the control loop adjusts the force that is output by each actuator. Thus the amount of force being applied at each joint is known. Additionally, each DOF is equipped with position sensors that are used for all levels of motion control and for maintaining the anti-gravity mode.

The vision system is responsible for detecting objects in the robot's field of view. A Gaussian mixture model is used to detect the background based on color, providing foreground/background classification. Connected regions with uniform color are extracted from the foreground regions. While this simple approach has several limitations, including the requirement that objects be of uniform color, as well as occasional errors due to shadows, it is sufficient for our current purposes. The output of the visual analysis module is a set of N connected regions. We denote the output of the visual system at time step t as $V[t] = \{R_1^v[t], R_2^v[t], ...R_N^v[t]\}$. Each region $R_i^v[t]$ consists of a list of member pixels and their RGB values. The vision system generates region vectors with an update rate of 15Hz. These region sets are passed to an object permanence module which integrates region sets over time to determine the presence and properties of objects in the scene.

The three-dimensional shape of an object is represented using a set of histograms, each of which represents the silhouette of the object from a different viewpoint [28]. We assume that with sufficient stored viewpoints, a novel viewpoint of an object may be matched by interpolation. A 2-D shape histogram is created using image pixels that correspond to an object based on figure-ground segmentation. First, all the outer edge points of the object are found. For each pair of edge points, two values are computed: the distance between the points, normalized by the largest distance between any two edge points, and the angle between the tangents to the edge of the object at the edge points. A 2-D histogram of all {distance, angle} values is accumulated for all pairs of edge points. The resulting histogram representation of the object silhouette is invariant to rotation (since all angles are relative) and object size (since all distances are normalized). Histograms are compared using the χ^2 divergence metric. Three dimensional shape is represented by bundling a set of 2-D histograms that represent different views of an object. This shape representation provides a basis for grounding words that refer to 3-D objects (e.g., *ball*).

The color of regions is also represented using histograms [28]. To compensate for lighting changes, the red (R), green (G), and blue (B) components of each pixel are divided by the sum of all three components ($R + G + B$) resulting in a set of "illumination-normalized" values. Since all triplets of illumination-normalized values must add to 1.0, there are only two free parameters for each pixel. For this reason, the normalized blue value of all pixels are not stored. A 2-D color histogram is generated by accumulating illumination-normalized red and green values for each pixel in the target region. The χ^2 divergence metric is also used to compare color histograms.

To enable grounding of spatial terms such as *above* and *left*, the set of three spatial features suggested in [17] is measured between pairs of objects. The first feature is the angle (relative to the horizon) of the line connecting the centers of area of an object pair. The second feature is the shortest distance between the edges of the objects. The third feature measures the angle (relative to the horizon) of the line which connects the two most proximal points of the objects.

The representations of shape, color, and spatial relations described above can also be generated from virtual scenes based on Ripley's mental model as described below. Thus, the visual features can serve as a means to ground words in either real-time, camera-grounded vision, or simulated synthetic vision.

D. Visually-Guided Reaching

Ripley can reach out and touch objects by interpolating between recorded motion trajectories. A set of sample trajectories are trained by placing objects on the tabletop, placing Ripley in a canonical position so that the table is in view, and then manually guiding the robot until it touches the object. A motion trajectory library is collected in this way, with each trajectory indexed by the position of the visual target. To reach objects in new positions, the appropriate linear interpolation between trajectories is computed.

E. Encoding Environmental Affordances: Object Weight and Compliance

Words such as *heavy* and *soft* refer to properties of objects that cannot be passively perceived, but require interaction with the object. Following Gibson [9], we refer to such properties of objects as *affordances*, highlighting what an object affords to an agent who interacts with it. For instance, a lightweight object can be lifted with ease as opposed to a heavy object. To assess the weight of an unknown object, an agent must actually lift (or at least attempt to lift) it and gauge the level of effort required. This is precisely how Ripley perceives weight. When an object is placed in Ripley's gripper, a motor routine tightly grasps the object and then lifts and lowers the object. While this motor program is running, the forces experienced in each DOF (Section III-B) are monitored. In initial word learning experiments, Ripley is handed objects of various weights and provided word labels such as *very light* or *heavy*. A simple Bayes classifier was trained to distinguish the semantics of *very light*, *light*, *heavy*, and *very heavy*. Similarly, we ground the semantics of *hard* and *soft* in terms of grasping motor routines that monitor pressure changes at each fingertip as a function of grip displacement.

IV. A Perceptually-Driven Mental Model

Ripley integrates real-time information from its visual and proprioceptive systems to construct an internal replica, or *mental model* of its environment that best explains the history of sensory data that Ripley has observed. The mental model is built upon the ODE rigid body dynamics simulator [29]. ODE enables modeling the dynamics of 3-D rigid objects based on Newtonian physics. As Ripley's physical environment changes, perception of these changes drives the creation, updating, and destruction of objects in the mental model. Although simulators are typically used in place of physical systems, we found physical simulation to be an ideal substrate for implementing Ripley's mental model (for other uses of coupled online simulation, see [30]–[32]). Our motivations for introducing a perceptually-aligned simulator were to provide:

- *Registration* Perceptual data provides evidence for determining when an object is believed to be "out there" in the objective world. When an object is determined to exist, it is instantiated in the internal mental model.
- *Tracking* Objects must be tracked across time to establish persistent referents for language.
- *Stabilization* Knowledge of Newtonian physics is used to enforce consistency of dynamics within the model. Early-stage visual processing errors that lead to physically impossible interpretations are overcome through stabilization in the simulator.
- *Compensating For Vantage Shifts* Compensation is necessary for shifts in physical point of view as Ripley mechanically moves its view, because movements in Ripley's body cause changes (often radical changes) in perspective. To accomplish this, a physical model of Ripley's own body is maintained in the internal model, and used to estimate the position of its cameras, thus providing a basis for perspective correction.
- *Enabling Virtual (Imagined) Shifts in Perspective* By moving the placement of a synthetic camera, Ripley is able to assume arbitrary points of view, including that of its human communication partner.
- *Event-based Memory* The mental model maintains a history of events, enabling Ripley to refer back in time.
- *Partial knowledge* Objects in the simulator may be represented with only partial knowledge of their properties through the use of confidence values.

The mental model mediates between perception of the objective world on one hand, and the semantics of language on the other. Although the mental model reflects the objective environment, it is biased as a result of a projection through Ripley's sensory system and built-in perceptual representations.

A. Physical Simulation

The ODE simulator provides facilities for creating and destroying rigid objects with arbitrary polyhedron geometries placed within a 3-D virtual world. Forces can be applied to move them, and other properties such as color and mass can be updated during simulation. Based on laws of Newtonian physics, ODE updates object positions at discrete time steps. Objects in Ripley's workspace (the tabletop) are constrained to be spheres of fixed size. Ripley's body is modeled within the simulator as a configuration of four connected cylindrical links terminated with a rectangular head that approximate the dimensions and mass of the physical robot.

B. Coupling Proprioception to the Mental Model

The model of the robot's body is controlled by a virtual motor controller that operates within the physical simulator. At each update cycle in the simulator, joint angles of the virtual robot are compared to the angles of the physical robot. For each joint, if the difference in angles is greater than a preset threshold, then an appropriate force is applied to align the corresponding virtual joint. In effect, the virtual joint tracks the associated DOF of the physical robot. Since only angle differences above a threshold lead to virtual forces, low level jitter in the physical robot are filtered in the simulator.

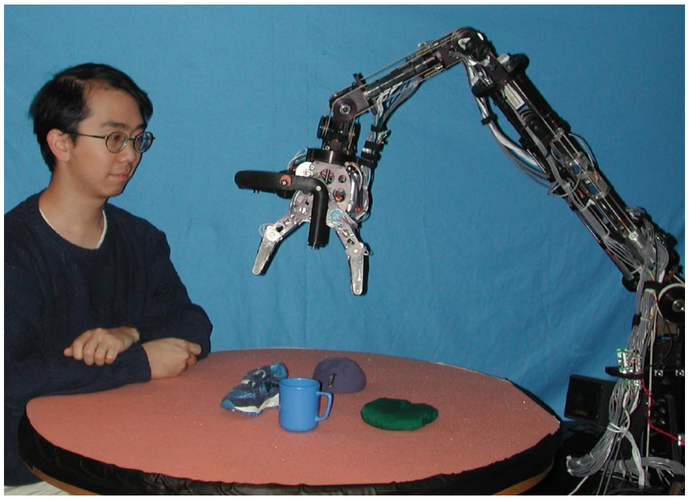

Fig. 3. Ripley looks down at the tabletop with four objects in view.

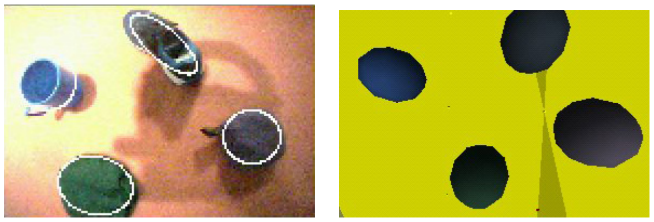

Fig. 4. Visual regions and corresponding simulated objects in Ripley's mental model corresponding to the view from Figure 3. The white ellipses in the left image indicate the output of the region analysis routines of the vision system. The objects in the simulator on the right are actually spherical, but appear elliptical due to optical warp of the synthetic viewpoint generated by the simulator.

C. Coupling Visual Perception to the Mental Model

A primary motivation for introducing the mental model was to register, stabilize, and track visually observed objects in Ripley's environment. An object permanence module, called the *Objecter*, was developed as a bridge between raw visual analysis and the physical simulator. When a visual region is found to stably exist for a sustained period of time, an object is instantiated by the Objecter in the ODE physical simulator. It is only at this point that Ripley becomes "aware" of the object and is able to talk about it. If Ripley looks away from an object such that the object moves out of view, a representation of the object persists in the mental model. When a physical object is removed from Ripley's workspace, persistent perceptual evidence of its disappearance will cause the object to be deleted from the model.

Figure 3 shows an example of Ripley looking over the workspace with four objects in view. In Figure 4, the left image shows the output from Ripley's head-mounted camera, and the right image shows corresponding simulated objects that have been registered and which are being tracked.

The Objecter consists of two interconnected components. The first component, the *2D-Objecter*, tracks two-dimensional visual regions generated by the vision system. The 2D-Objecter also implements a hysteresis function which detects visual regions that persist over time. The second component, the *3D-Objecter*, takes as input persistent visual regions from the 2D-Objecter, which are brought into correspondence with a full 3-D physical model which is maintained by ODE. The 3D-Objecter performs projective geometry calculations to approximate the position of objects in 3-D based on 2-D region locations combined with the position of the source video camera (i.e., the position of Ripley's head). As Ripley moves (and thus changes its vantage point), the 2D-Objecter continues to track visual regions until they leave the field of view. However, updates to the 3-D mental model are not performed while Ripley is in motion. While this assumption simplifies the process of tracking objects, it will be relaxed in the future work. Both the 2-D and 3-D Objecter maintain correspondence of objects across time, enabling tracking and object persistence in spite of perceptual gaps and noise.

Recall from Section III-C that the output of the vision system at each time step is a set of N visual regions, $V[t] = \{R_1^v[t], R_2^v[t], \ldots, R_N^v[t]\}$. In general, the ordering of regions within V is arbitrary since the vision system finds regions in each frame of video independent of knowledge of previous frames. Thus, there is no guarantee that $R_i^v[t]$ will correspond to $R_i^v[t+1]$.

To obtain correspondence of regions over time, the 2D-Objecter maintains its own set of regions which are candidates for being output to the 3D-Objecter. We denote the candidate region set as $O[t] = \{R_1^o[t], R_2^o[t], \ldots, R_M^o[t]\}$. In contrast to V, the purpose of the 2D-Objecter is to maintain correspondence between $R_i^o[t]$ and $R_i^o[t]$. Each candidate region has an associated confidence value $R_i^o[t].conf$.

A tunable distance metric between two visual regions is defined as:

$$d(R_i, R_j) = \alpha d_p(R_i, R_j) + \beta d_s(R_i, R_j) + (1-\alpha-\beta) d_c(R_i, R_j) \quad (1)$$

Where $d_p()$ is the Euclidean distance between the centroids of the regions, $d_s()$ is the difference in size (number of pixels) of regions, and $d_c()$ is the difference in average RGB color of the regions. The tuning parameters α and β are scalar values such that $(\alpha + \beta) \leq 1$. They are used to set the relative emphasis of the position, size, and color properties in comparing regions.

When Ripley moves to a new vantage point, the 2D-Objecter candidates are initialized by copying the output of the vision system ($O \leftarrow V$) so that a candidate is created corresponding to each region in the current visual analysis frame. The confidence of each candidate is set to 0. At each successive time step, a new region set is generated by the vision system. The 2D-Objecter attempts to put each region in V into one-to-one correspondence with each candidate in O such that the total distance between paired regions is minimized. In general, the number of visual regions N and 2D-Objecter candidate regions M will not be equal. The alignment process aligns the $min(N, M)$ subset of regions. However, after the optimal alignment is found, only those whose distances resulting from the match that are below a maximum allowable distance threshold are accepted. The confidence of candidate regions that are aligned to regions from V is updated (increased) using a rule similar to an infinite impulse response filter, assuming positive input of unit magnitude. Thus, effectively, confidence values never reach an upper bound of one. If $N > M$, at most $(N - M)$ new candidates are instantiated in the 2D-Objecter, each with confidence set to 0. If $N < M$, then the confidence of the at least $(M - N)$ unaligned candidate regions is updated (decreased) by a similar rule, driven by a negative input of unit magnitude. At the end of this alignment and confidence update process, the properties of the matched or newly instantiated regions from S_v are copied into R^c. The unmatched candidate regions retain their previous properties, and any of them for which $R_i^c[t].conf < 0$ are destroyed.

The output of the 2D-Objecter at each time step is the subset of candidate regions for which the confidence level is greater than $Conf_{MIN}$. In the current implementation, $Conf_{MIN} = 0.9$. Each newly instantiated candidate region is assigned a unique ID. These IDs are persistent over time, thus implementing region tracking. Smoothly moving objects are tracked by the 2D-Objecter. When an object is removed from a scene, the confidence value of the corresponding candidate region will drop steadily from a maximum value of $Conf_{MAX}$. As soon as the confidence drops below $Conf_{MIN}$, it stops being output. This use of confidence values and thresholds implements a hysteresis function that requires persistent visual evidence before either instantiating or destroying regions.

The 3D-Objecter uses projective geometry to infer the position of objects in three-dimensional space based on 2D regions. Given the position and orientation of Ripley's camera (which are derived from the body model described in Section IV-B), the 2-D regions are linearly projected in 3-D until the projection lines intersect Ripley's work surface. The location of the surface, a round tabletop, is built into the ODE simulator. Thus, Ripley's perceptual input is not necessary for establishing the presence of the table.

Interaction between the 2D- and 3D-Objecter proceeds as follows. Each time Ripley moves, the 3D-Objecter ignores output from the 2D-Objecter, and when Ripley stabilizes its position, the 3D-Objecter waits 0.5 seconds to ensure that the 2D-Objecter's region report is stable, and then resumes 3-D processing. When the 3D-Objecter processes a 2-D region set, it projects each region to a corresponding 3-D object location. Then, the projected objects are then placed into correspondence with existing objects in ODE. To compare projected and existing objects, a modified version of Equation 1 is used in which $d_p()$ measures three dimensional Euclidean distance, and $d_s()$ is not computed (since the current version of the simulator assumes all objects to be of equal size). The same alignment process as the 2D-Objecter is used to align projected objects to existing objects in ODE. If projected objects have no existing counterparts in the simulator, new objects are instantiated. Conversely, if an object exists in ODE but no corresponding object has been projected based on visual evidence, then the object in ODE is destroyed. There is no hysteresis function required in the 3D-Objecter since all 2-D regions have already passed through a hysteresis function in the 2D-Objecter.

D. Dynamics in the Mental Model: Inference of Force Vectors

In the process of updating the position of moving objects, the 3D-Objecter must infer the magnitude and direction of forces which lead to observed motions. Inference of force dynamics has been argued to be of fundamental importance in grounding verb meanings [33], a direction we will pursue in the future. Here, we explain how forces are inferred from visual observation in the Objecter.

Consider a situation in which an object, such as a ball, is on the workspace and in view. Once the 2D-Objecter has registered the corresponding region, it will relay the region to the 3D-Objecter which will instantiate an object in ODE. At this point, Ripley is aware of the ball. Now, the ball begins to slowly roll. Although the visual region corresponding to the ball will be displaced from one time step to the next, the 2D-Objecter will generally determine the correspondence between regions over time steps and thus track the object. After the correspondence process has been run by the 3D-Objecter, a displacement in positions between projected and existing objects in the simulator must be accounted for. This is where the force inference step takes place. A force proportional to the displacement and in the direction of the projected object is applied within ODE to the corresponding object. As the object (de)accelerates, the inferred forces will be de/increased accordingly. To summarize, in the process of tracking objects, the Objecter also generates a constant stream of inferred forces acting on each object to account for their changes in velocity. These force vectors may be used to classify self-moving objects, and other aspects of force dynamics.

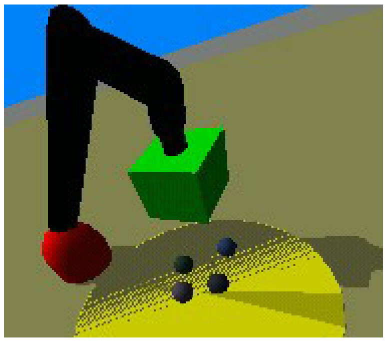

Fig. 5. By positioning a synthetic camera at the position approximating the human's viewpoint, Ripley is able to "visualize" the scene from the person's point of view, which includes a view of Ripley.

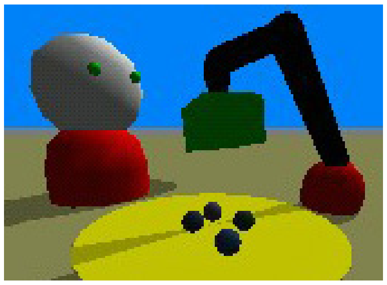

Fig. 6. Using virtual shifts of perspective, arbitrary vantage points may be taken. The (fixed) location of the human partner is indicated by the figure on the left.

E. Synthetic Vision and Imagined Changes of Perspective

The ODE simulator is integrated with a 3-D graphics rendering environment [34]. The 3-D environment may be rendered from an arbitrary viewpoint by positioning and orienting a synthetic camera and rendering the scene from the camera's perspective. Changes in placement of the synthetic camera are used to implement shifts in perspective without physically moving Ripley. Figures 5 and 6 show examples of two synthetic views of the situation also depicted in Figures 3 and 4. The visual analysis features described in Section III-C can be applied to the images generated by synthetic vision. We will return to this as the mechanism for grounding *my left* as a combination of a shift of perspective that modulates the frame of reference of a spatial relation model.

F. Event-Based Memory

To support linguistic references to the past, memory must be added to the mental model. Although all details of mental model activity can be archived verbatim, humans do not actually encode every detail of experience, but instead retain only important or salient aspects of the past. What counts as salient or important is an immensely difficult problem relating to questions of attention, goals and so forth. We present a simple model of memory encoding useful for representing past events in ways congruent with how people are likely to refer to them in natural language.

Run length encoding is used to compactly represent mental model histories. Each time an object changes a property by more than a set threshold using the distance measure in Equation 1 (modified for 3-D simulated objects), an event is detected in the mental model and recorded in memory. Time periods during which no significant changes occur are collapsed in memory. When an object changes properties, such as its position, only the beginning and end points of the change are retained. As a result, references to the past are discretized in time along event boundaries.

V. Putting the Pieces Together: Foundations for Grounding Conversational Language

We began by asking how a robot might ground the meaning of the utterance, *Touch the heavy blue thing that was on my left*. We are now able to sketch an answer to this question based on a set of sensory, motor, and simulation-grounded representations that have been described in the preceding sections.

The semantic grounding of each word in our example utterance is defined using pseudo-algorithmic functions reminiscent of the procedural semantics developed by Winograd [35] and Miller & Johnson-Laird [36]. These functions ground lexical semantics in terms of procedures and representations that are implemented in Ripley. Rather than specify complete details of how the words would be grounded (which must await future implementation of a language processing module for Ripley), we show how the various architectural features of the robotic architecture might come together in grounding each word of the utterance.

We begin by introducing a data structure called $context$ that contains two components:

```
context {
  point-of-view
  working-memory
}
```

By default, the point-of-view is set to Ripley's first-person perspectve, and the contents of working memory are set to be the contents of the mental model. As we shall see, semantic procedures are able to modulate the contents of $context$.

The meaning of the word *blue* may be defined as:

blue(x): f_{blue}(GetColorModel(x))

Where $f_{blue}()$ encodes an expected region of color space using an appropriately tuned color histogram (Section III-C), and GetColorModel() returns a color histogram of the object x:

```
GetColorModel(x) {
  if model exists in working memory then
    fetch and return
  else
    LookAt(x)
    return (BuildColorModel(x))
  end if
}
```

$GetColorModel(x)$ first checks working memory, and if the color model is not built, it causes the robot to look at the object x and return a model. The resulting color model is also stored in working memory. In effect, the semantics of *blue* are grounded not only in the visual system, but also Ripley's motor system through the $LookAt()$ function. The grounding occurs by virtue of the direct connections between each function and Ripley's sensory-motor system; $f_{blue}()$ specifies expected values of pixels captured by Ripley's cameras; $LookAt()$ specifies control procedures for Ripley's actuators to bring an object into view.

We can ground *heavy* in much the same way as *blue*:

heavy(x): f_{heavy}(GetWeight(x))

Where $f_{heavy}()$ encodes an expected region of one-dimensional weight space using an appropriately tuned probability density function, and $GetWeight()$ returns the weight of x:

```
GetWeight(x) {
  if weight known in working memory then
    fetch and return
  else
    return (weigh(x))
  end if
}
```

$weigh()$ corresponds to the functionality described in Section III-E and may be defined here as:

```
weigh(x) {
  grasp(x)
  resistance $\leftarrow$ 0
  while lift(x) do
    resistance $\leftarrow$ resistance + joint forces
  end while
  return resistance
}
```

The grounding of *heavy* mirrors that of *blue* in the use of a tuned function (f_{heavy} / f_{blue}). These tuned functions represent sensory expectations, each of which are linked to appropriate motor functionality (lifting / looking).

In general, the use of *heavy* must be scaled dependent on context (e.g., a heavy chair is lighter than a light car). Various schemes including within-category rescaling, and within-working memory rescaling might be used. Similar rescaling must performed on other modifier terms including color (consider the difference in color models for *red* when applied to paint, wine, hair, wood, and skin).

Touch can be grounded in the perceptually-guided motor procedure described in Section III-D. This reaching gesture terminates successfully when the touch sensors are activated and the visual system reports that the target x remains in view (as it should given Ripley's head-mounted camera):

```
touch(x) {
  repeat
    ReachTowards(x)
  until touch sensor(s) activated
  if x in view then
    return success
  else
    return failure
  end if
}
```

Left is grounded in a visual property model which computes a geometric spatial function (Section III-C) relative to the assumed point of view:

```
left(x) {
  trajector $\leftarrow$ GetPosition(x)
  return $f_{left}$(trajector, point-of-view)
}
```

$GetPosition()$, like $GetColorModel()$, would use the least effortful means for obtaining the position of x. The function $f_{left}()$ evaluates how well the position of x fits a spatial model relative to the point of view determined from $context$.

Thing can be grounded as a pair of affordances:

thing(x): IsTouchable(x) and IsViewable(x)

This definition states that for x to be a thing, it must be touchable and viewable. Touchability would be grounded using $Touch()$ and viewability based on whether x has appeared in the mental model based on visual perception.

Was triggers a transfer of contents from the mental model memory (Section IV-F) into working memory, making them accessible to other processes.

was(context): working memory $\leftarrow$ mental model history

My triggers a shift in spatial perspective and may thus be grounded in the synthetic visual perspective shift operation described in Section IV-E:

my(context): point-of-view $\leftarrow$ GetPointOfView(speaker)

Where $GetPointOfView(speaker)$ obtains the spatial position and orientation of the speaker. In the current implementation, the position of the speaker is set to a fixed location in Ripley's mental model (i.e., at a fixed location and orientation in the ODE simulator). In general, however, various sensing methods such as face detection and sound localization might be used to dynamically ascertain the point of view of the speaker. The function of $my()$ is to shift the point of view in $context$ to assume the speaker's perspective.

The determiner *the* indicates the selection of a single referent from working memory, which in turn is constructed from Ripley's mental model and memory system:

the(context): Select most salient item from working memory

In summary, we have shown how each word in the sample sentence can be grounded in Ripley's sensory-motor and mental model representations and processes. All primitives used to ground the words build on perceptual, motor, and simulation processes of Ripley, along with simple bookkeeping using the $context$ data structure.

The final step in interpreting the utterance is to compose the semantics of the individual words in order to derive the semantics of the whole utterance. We assume that a syntactic parser is able to parse the utterance and translate it into a nested set of function calls[2]:

Touch(The(Left(My(Heavy(Blue(Thing(Was(context))))))))

Consider how this nested set of functions might be interpreted. The innermost argument is $context$ which includes the assumed point of view and contents of working memory. Each nested function call modifies the contents of $context$ by either shifting points of view, loading new contents into working memory, or sorting / highlighting contents of working memory. $Touch()$ finally acts on the specified argument to execute the command.

Starting from the innermost function call, $was()$ loads partial history from the mental model into working memory. $Thing()$ selects the subset of elements in working memory which possess the affordances $IsTouchable()$ and $IsViewable()$. In operation, obtaining trustworthy values for these affordances may be impractical. For every visible object, Ripley would have to reach out and touch the object to make sure it $IsTouchable()$ and thus not an optical illusion. In lieu of this costly exercise, the affordances may be assigned default values. For instance, a reasonable rule is that three-dimensionally visible objects that persist over time should by default be considered $IsTouchable()$. If during attempted manipulation this turns out to be false, that knowledge would be updated in Ripley's knowledge base[3].

The next two functions, $blue()$ and $heavy()$, sort objects in Ripley's working memory according to how well they match the property models of each function. Objects that receive very poor evaluations by either property function would be removed from further consideration. $My()$ updates the point of view in $context$ to take the speaker's point of view. $Left()$ sorts objects based on the frame of reference supplied by $context$. $The()$ selects the single most salient object from working memory. Finally, $Touch()$ is executed, causing Ripley to touch the specified object. The precise sequence of function nesting is this example is not critical (e.g., $blue()$ and $heavy()$ could be interchanged without effect, $my()$ can be applied at any point before $left()$ but does not need to be adjacent to it, etc.).

The semantic procedures as spelled out here are skeletal in nature and will require further refinements as our implementation proceeds. Although the strict use of a procedural notation forces a linear, recipe-like definition of word meaning, the notation serves our current expository purpose for demonstrating how elements of the architecture provide a basis for language grounding. Our intent has been to convey an overall gist of how language would be coupled to Ripley. Our current work is focused on the realization of this approach using spoken language input.

[2]We address the problem of grounded semantic composition in detail elsewhere [37].

[3]This is similar to Minsky's suggestion for frame based representations in which slots can have weakly attached default values which can be overridden based on contrary evidence [38]

VI. Conclusions

We believe that the best way to build a conversational robot is to take a holistic approach to semantics in which the robot's perception, motor control, memory, and planning mechanisms are all seamlessly tied to linguistic representations. We have described a robotic architecture that provides contextually-derived perceptual, procedural, and affordance representations suitable for grounding word meanings. First steps towards grounding social and temporal words such as *my* and *was* have been taken through the use of a perceptually-coupled mental model and memory system. Ongoing work includes implementation of a conversational module for Ripley, and further development of physical and social representations in service of grounding word meaning.

Acknowledgments

Ben Krupp and Chris Morse implemented and helped with modifications to the hardware components of Ripley. Niloy Mukherjee implemented the visual object detection algorithms used in Ripley.

References

[1] D. Bailey, "When push comes to shove: A computational model of the role of motor control in the acquisition of action verbs," Ph.D. dissertation, Computer science division, EECS Department, University of California at Berkeley, 1997.

[2] S. Narayanan, "Karma: Knowledge-based active representations for metaphor and aspect," Ph.D. dissertation, University of California Berkeley, 1997.

[3] T. Regier and L. Carlson, "Grounding spatial language in perception: An empirical and computational investigation," *Journal of Experimental Psychology*, vol. 130, no. 2, pp. 273–298, 2001.

[4] D. Roy and A. Pentland, "Learning words from sights and sounds: A computational model," *Cognitive Science*, vol. 26, no. 1, pp. 113–146, 2002.

[5] J. Siskind, "Grounding the Lexical Semantics of Verbs in Visual Perception using Force Dynamics and Event Logic," *Journal of Artificial Intelligence Research*, vol. 15, pp. 31–90, 2001.

[6] J. M. Lammens, "A computational model of color perception and color naming," Ph.D. dissertation, State University of New York, 1994.

[7] L. Steels, "Language games for autonomous robots," *IEEE Intelligent Systems*, vol. 16, no. 5, pp. 16–22, 2001.

[8] P. McGuire, J. Fritsch, J. Steil, F. Roethling, G. Fink, S. Wachsmuth, G. Sagerer, and H. Ritter, "Multi-modal human-machine communication for instructing robot grasping tasks," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2002.

[9] J. J. Gibson, *The Ecological Approach to Visual Perception*. Erlbaum, 1979.

[10] J. Barwise and J. Perry, *Situations and Attitudes*. MIT-Bradford, 1983.

[11] G. Lakoff and M. Johnson, *Metaphors We Live By*. Chicago: University of Chicago Press, 1980.

[12] G. Miller, "Wordnet: A lexical database for english," *Communications of the ACM*, vol. 38(11), pp. 39–41, 1995.

[13] D. Jurafsky and J. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Speech Recognition, and Computational Linguistics*. Prentice-Hall, 2000.

[14] M. K. Brown, B. M. Buntschuh, and J. G. Wilpon, "SAM: A perceptive spoken language understanding robot," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22 . IEEE Transactions 22, pp. 1390–1402, 1992.

[15] D. Perzanowski, A. Schultz, W. Adams, K. Wauchope, E. Marsh, and M. Bugajska, "Interbot: A multi-modal interface to mobile robots," in *Proceedings of Language Technologies 2001*, Carnegie Mellon University, 2001.

[16] C. Crangle and P. Suppes, *Language and Learning for Robots*. Stanford, CA: CSLI Publications, 1994.

[17] T. Regier, *The human semantic potential*. Cambridge, MA: MIT Press, 1996.

[18] D. Roy, “Learning words from sights and sounds: A computational model,” Ph.D. dissertation, Massachusetts Institute of Technology, 1999.

[19] B. Landau and R. Jackendoff, “”What” and ”where” in spatial language and spatial cognition,” *Behavioral and Brain Sciences*, vol. 16, pp. 217–265, 1993.

[20] H. Clark, *Using Language*. Cambridge University Press, 1996.

[21] P. Bloom, *How Children Learn the Meanings of Words*. Cambridge, MA: MIT Press, 2000.

[22] C. Breazeal, “Towards sociable robots,” *Robotics and Autonomous Systems*, vol. 42, no. 3-4, 2003.

[23] B. Scassellati, “Theory of mind for a humanoid robot,” *Autonomous Robots*, vol. 12, pp. 13–24, 2002.

[24] C. Sidner and M. Dzikovska, “Hosting activities: Experience with and future directions for a robot agent host,” in *ACM International Conference on Intelligent User Interfaces*, 2002, pp. 143–150.

[25] T. Paek and E. Horvitz, “Conversation as action under uncertainty,” in *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, 2000.

[26] J. Pratt, B. K. B, and C. Morse, “Series elastic actuators for high fidelity force control,” *Industrial Robot*, vol. 29, no. 3, pp. 234–241, 2002.

[27] F. Mussa-Ivaldi and E. Bizzi, “Motor learning through the combination of primitives,” *Philosophical Transactions of the Royal Society of London*, vol. 355, pp. 1755–1769, 2000.

[28] D. Roy, B. Schiele, and A. Pentland, “Learning audio-visual associations from sensory input,” in *Proceedings of the International Conference of Computer Vision Workshop on the Integration of Speech and Image Understanding*, Corfu, Greece, 1999.

[29] R. Smith, “ODE: Open dynamics engine,” 2003. [Online]. Available: http://q12.org/ode/

[30] F. Cao and B. Shepherd, “Mimic: a robot planning environment integrating real and simulated worlds,” in *IEEE International Symposium on Intelligent Control*, 1989, p. 459464.

[31] W. J. Davis, “On-line simulation: Need and evolving research requirements,” in *Handbook of Simulation: Principles, Methodology, Advances, Applications and Practice*, J. Banks, Ed. Wiley, 1998.

[32] J. R. Surdu, “Connecting simulation to the mission operational environment,” Ph.D. dissertation, Texas A&M, 2000.

[33] L. Talmy, *Toward a Cognitive Semantics*. Cambridge, MA: MIT Press, 2000.

[34] “OpenGL.” [Online]. Available: www.opengl.org

[35] T. Winograd, *A Process model of Language Understanding*. Freeman, 1973, pp. 152–186.

[36] G. Miller and P. Johnson-Laird, *Language and Perception*. Harvard University Press, 1976.

[37] P. Gorniak and D. Roy, “Grounded semantic composition for visual scenes,” forthcoming, 2003.

[38] M. Minsky, “A framework for representing knowledge,” in *The Psychology of Computer Vision*, P. Winston, Ed. New York: McGraw-Hill, 1975, pp. 211–277.