Where Do I Look Now? Gaze Allocation During Visually Guided Manipulation

Jose Nunez-Varela, B. Ravindran, Jeremy L.Wyatt

***Abstract*— In this work we present principled methods for the coordination of a robot's oculomotor system with the rest of its body motor systems. The problem is to decide which physical actions to perform next and where the robot's gaze should be directed in order to gain information that is relevant to the success of its physical actions. Previous work on this problem has shown that a reward-based coordination mechanism provides an efficient solution. However, that approach does not allow the robot to move its gaze to different parts of the scene, it considers the robot to have only one motor system, and assumes that the actions have the same duration. The main contributions of our work are to extend that previous reward-based approach by making decisions about where to fixate the robot's gaze, handling multiple motor systems, and handling actions of variable duration. We compare our approach against two common baselines: random and round robin gaze allocation. We show how our method provides a more effective strategy to allocate gaze where it is needed the most.**

I. INTRODUCTION

A. Problem Definition

Real-world tasks require robots to act under uncertain and incomplete information. In this work we show how a robot should act to reduce that uncertainty by controlling its gaze in a principled way. We consider robots that have an oculomotor system and multiple motor systems. The oculomotor system can move the robot's cameras to specific fixation points. After a fixation is made, the robot can extract visual information from the captured image. The robot's motor systems (e.g. its arms) can perform physical actions (e.g. grasping an object), and can run simultaneously. Furthermore, at any point in time a particular motor system might require the oculomotor system to gather information relevant to the success of its physical actions.

To achieve this a coordination mechanism must be defined in order to allocate the control of gaze to the motor system which most needs it. Since the camera has a limited field of view not all different motor systems can be assisted by a single fixation. Thus the robot needs to choose which motor system to assist with gaze. Also, visual information is noisy, so the robot should be able to handle uncertainty.

Our coordination framework has been implemented using the iCub simulator [1] (Fig. 1). The iCub is a humanoid robot with multiple motor systems and an oculomotor system. The iCub simulator is a powerful tool that simulates closely what the real robot can do, it uses exactly the same controllers and modules used to control the real robot. The main advantage of using a simulator is to have greater control over the configuration of the environment and the robot. For instance, the creation and manipulation of objects, and changing its field of view can be altered easily between trials.

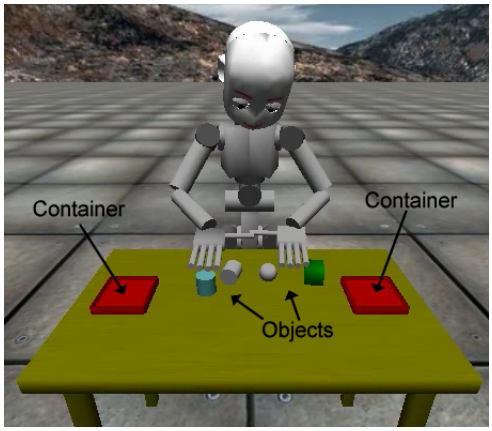

Fig. 1. Snapshot of the iCub Simulator. The task is to pick up and place objects from the table top to the containers.

To test our approach we have defined a task that consists of picking up objects from the table top and then placing them inside one of two containers. In this paper, the arms cannot interact with each other (e.g. to perform bi-manual grasps), so the task is divided into two sub-tasks, one for each arm. This means that objects reachable by the right arm are placed inside the right container, with the same happening for the left side. A new object appears on the table every 60 seconds and also every time an object is put inside a container. The robot has to look at objects and containers in order to get an estimate of their location. The only precisely known location to the robot is the centre of the table.

B. Related Work

Current research on visual perception has focused on attentional systems based on saliency [2] in order to detect regions of interest which may provide possible fixation points. Some other systems incorporate saliency and active vision [3], that allow the robot to move its camera to get different views of one object. Although these systems provide a good way to predict where to direct the gaze, they normally fail to incorporate information about the task being performed. In

We gratefully acknowledge the support given by CONACYT-Mexico (Reg.179604), UKIERI (SA06-0031), CogX (FP7-ICT-215181).

J. Nunez-Varela is with the School of Computer Science, University of Birmingham, B15 2TT Birmingham, UK j.i.nunez@cs.bham.ac.uk

B. Ravindran is with the Department of Computer Science and Engineering, IIT Madras, India ravi@cse.iitm.ac.in

J. Wyatt is with the School of Computer Science, University of Birmingham, B15 2TT Birmingham, UK j.l.wyatt@cs.bham.ac.uk

[4] a system capable to plan its image processing actions is presented, but the camera remains fixed to a single position.

Sprague and Ballard [5] developed a reward-based perceptual coordination mechanism for a simulated human-agent. The agent performs a set of behaviours concurrently (where each behaviour has a separate goal), by sharing the set of actions amongst them. Each behaviour also has an associated visual routine that updates the information used by that behaviour. Only one visual routine can be executed each time step. Their key idea is to select the visual routine which minimises uncertainty for the behaviour that is likely to lose more reward. Their results show that a reward-based approach provides an effective way to coordinate perception and action. However, there are several restrictions in their approach. First, the agent's eyes do not move independently from its body. Second, the agent has only one motor system that is shared by all the behaviours. Third, they require that all physical actions have the same duration.

C. Contributions

We extend Sprague and Ballard's approach by:

- Considering a separate oculomotor system for control of the robot's cameras.
- Considering multiple motor systems.
- Considering actions with variable duration.
- Considering the actual information to be gained by perceptual actions rather than assuming all uncertainty disappears in one gaze.

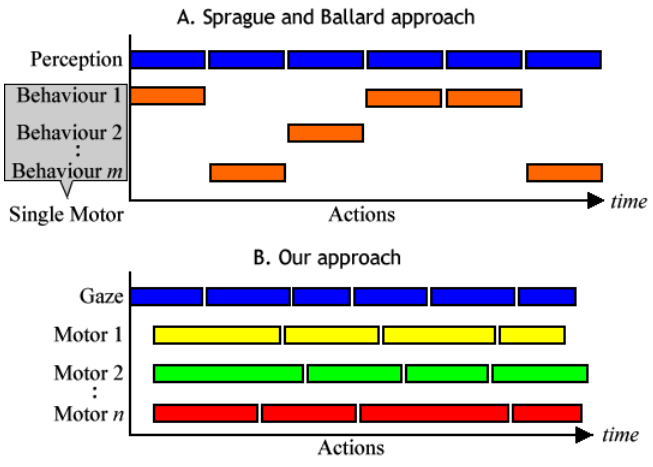

Fig. 2. A. Sprague and Ballard's approach handles several concurrent behaviours but only one motor system with actions of identical duration. B. Our approach handles multiple motor systems operating in parallel with actions of variable duration. Each block represents an action.

Fig. 2 shows a graphical representation of Sprague and Ballard's system handling concurrent behaviours with a shared motor system and with actions of the same duration. In contrast, our system considers multiple motor systems (where each motor system achieves a single goal), and actions of variable duration.

D. Modelling Decision Problems

A common approach for modelling decision making problems is to use Markov decision processes (MDPs) [6]. Sprague and Ballard model each behaviour as an MDP, which they use to learn the agent's task via reinforcement learning (RL) [7]. Their perceptual coordination mechanism is formalised as a partially observable MDP (POMDP) [8], in order to handle the state uncertainty during execution time. The selection of perceptual actions is done by a modified version of the *Q-MDP algorithm* [9]. This algorithm is a one-step look-ahead strategy for solving POMDPs that assumes the current uncertainty will disappear after executing the current action. They also use Kalman filters [10] in order to maintain an estimate of the state variable and its uncertainty.

One disadvantage with MDPs is the assumption that actions have the same duration. In real-world tasks, actions can take variable amounts of time to finish (e.g. the time for reaching and grasping an object is likely to be different). Thus, we model our problems using semi-MDPs (SMDPs) [6], that allow us to model actions with variable duration. Furthermore, we follow the concept of *options* [11], defined as temporally extended actions that can be composed of one or multiple single-step actions.

By having multiple motor systems it is possible to exploit parallelism and/or concurrency. In this paper, we take into account only parallelism (i.e. motor systems do not interact with each other). Thus, we decompose the task and model each motor system as a separate SMDP, then each of them learns its corresponding sub-task via RL. By modelling the task using options we end up with a two-level state space. The high-level state space describes the task qualitatively, and the lower-level state space describes continuous variables representing uncertainty. We make use of particle filters [10] to keep track of the state variables and its uncertainty (in our case the location of objects and containers). We have chosen particle filters over Kalman filters because the probability distributions modelled do not need to be Gaussians, thus allowing more flexibility. Section II describes the theory behind our proposed framework. Section III presents the experiments and results, and Section IV provides some final remarks and future work.

II. Coordinating Gaze and Actions

In this section we first describe how the task is modelled, then we explain the different components of our system and their interaction as Fig. 3 illustrates.

A. Modelling the task

As described in Section I, the task is to pick up objects from the table top and then place them inside one of two containers. We consider two motor systems: the right and left arm/hand. Thus we divide the task into two sub-tasks, one for each motor system. Furthermore, each sub-task is modelled as a hierarchy of two levels. The high-level models the sub-task qualitatively as an SMDP with a discrete state representation. Whereas the low-level models each high-level action with a continuous state space representation.

TABLE I

FACTORISED STATE SPACE FOR RIGHT AND LEFT ARM.

State Variable	*State Value*
armPosition	{onObject, onTable, onContainer, outsideTable}
handStatus	{grasping, empty}
tableStatus	{objectsOnTable, empty}

TABLE II

OPTIONS FOR RIGHT AND LEFT ARMS.

Options	*Average completion time (secs)*
moveToObject	2.65
moveToTable	2.95
moveToContainer	3.25
graspObject	2.87
releaseObject	1.0

In the high-level SMDP, each motor system $ms \in MS$ (where MS is the set of motor systems), is modelled as a tuple $\langle \mathcal{S}_{ms}, \mathcal{O}_{ms}, \mathcal{T}_{ms}, \mathcal{R} \rangle$, where $\mathcal{S}_{ms}$ is the set of discrete states, $\mathcal{O}_{ms}$ is the set of *options* (temporally extended high-level actions [11]), $\mathcal{T}_{ms} : \mathcal{S}_{ms} \times \mathcal{O}_{ms} \times \mathcal{S}_{ms} \times \mathbb{N} \rightarrow [0, 1]$ is the transition probability distribution, where $\mathbb{N}$ is the set of natural numbers representing the execution time of each option, and $\mathcal{R} : \mathcal{S}_{ms} \times \mathcal{O}_{ms} \rightarrow \mathbb{R}$ is the reward function. For this domain, the reward function is identical for all motor systems.

Typically, an option $\mathcal{O}^j$ is modelled as $\mathcal{O}^j = \langle \mathcal{M}^j, \mathcal{I}^j, \beta^j \rangle$, where $\mathcal{M}^j = \langle \mathcal{S}^j, \mathcal{A}^j, \mathcal{T}^j, \mathcal{R}^j \rangle$ is an MDP, $\mathcal{I}^j \subseteq \mathcal{S}^j$ is the initiation set where the option can start, and $\beta^j : \mathcal{S}^j \rightarrow [0, 1]$ defines a termination condition. In our case, an option $\mathcal{O}^j_{ms} \in \mathcal{O}_{ms}$ is modelled with continuous states ($\mathcal{S}^j$) and actions ($\mathcal{A}^j$), and with a non-stochastic transition function ($\mathcal{T}^j$). Options are defined as commands provided by the motor controllers available to the iCub [12]. These controllers receive 3D positions in robot coordinates and they calculate and execute trajectories in the joint space of the robot.

The set of motor systems for the high-level SMDPs is defined as $MS = \{right\_arm, left\_arm\}$. Table I shows the factorised discrete state space used for both arms ($\mathcal{S}_{right\_arm}$ and $\mathcal{S}_{left\_arm}$). Table II defines the set of options available for both arms ($\mathcal{O}_{right\_arm}$ and $\mathcal{O}_{left\_arm}$), and next to each option is the average completion time in seconds. These times were obtained by executing all options in sequence for 60 minutes. It is important to point out that for option *graspObject* we use a special command, defined in the iCub simulator, that makes the hand act like a magnet. Even though we simplify the problem of grasping, we can control its sensitivity by checking the offset between the centre of the hand and the centre of the object. Except for *releaseObject*, all options can fail if the offset between the centre of the hand and the desired final position is greater than some threshold (e.g. 1 cm). The iCub controllers have a limited accuracy and the minimum value of that threshold is 0.5 cm.

B. Visual Memory

The central component in the system is what we call the *visual memory* (Fig. 3), which captures the continuous state information about object pose needed for low-level control and supplies the discrete objects id's needed to create the high-level discrete state. We define the visual memory as the set of ordered pairs $\mathcal{VM} = \langle (e_1, bel(e_1)), (e_2, bel(e_2)), \ldots, (e_n, bel(e_n)) \rangle$, where e_i is the i^{th} entity of interest on the table, where an entity can be an object or a container. Every time a new entity is seen it is added to the visual memory. $bel(e_i)$ is a probability density (or belief state), associated with the location of the entity e_i. This location is used by the low-level options. Each object has a diameter of 4 cm, and its location refers to its centre projected in the X-Y plane in robot coordinates. Containers are 10x10x3 cm in width, length and height, and the location refers to its centre as well. In our case, the belief state for each entity e_i will be approximated by a particle filter, where each particle represents a possible location. In this paper we distinguish between the belief state for an individual entity, the overall belief state for the task (i.e. the set of belief states for all e_i), and the belief state relevant to a particular motor system ms (i.e. the set of belief states for all e_i that are relevant to motor system ms at the current time).

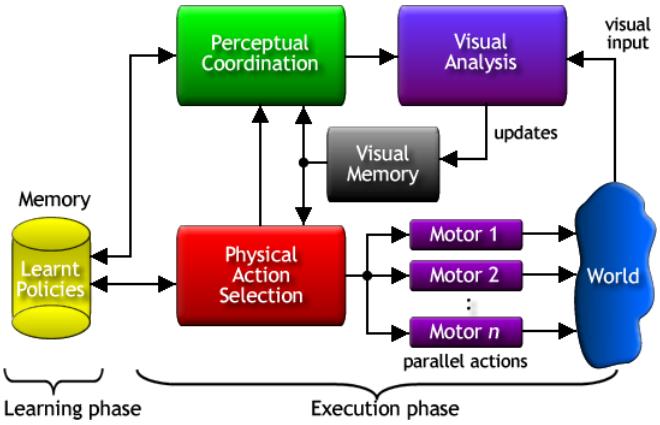

Fig. 3. Interaction between the components of the system.

C. Learning Phase

As described above, each high-level SMDP models a given sub-task, and learning this sub-task is achieved via RL using *SMDP Q-learning* [13]. Each motor system ms should learn a policy $\pi_{ms} : \mathcal{S}_{ms} \rightarrow \mathcal{O}_{ms}$, that defines a mapping from states to options. The robot initially learns how to perform the task under an assumption of complete observability. This means that the visual memory contains the complete list of entities (e_i), and their corresponding belief ($bel(e_i)$) indicates the true location of that entity. We follow a minimal time to goal strategy, so for any option taken the robot receives -1 unit of reward. When the task is completed (i.e. there are no objects on the table), it receives 0 units of reward. It receives -2 units if it tries to grasp an object when the hand is not empty, when the hand is not on an object, and if it tries to

release an object when the hand is empty. For our task, the policy learnt for one arm can be used for the other as well. The learning rule for SMDP Q-learning is formulated as:

$$\mathcal{Q}(s,o) \leftarrow \mathcal{Q}(s,o) + \alpha \left[r + \gamma^k \max_{o' \in \mathcal{O'}_{ms}} \mathcal{Q}(s',o') - \mathcal{Q}(s,o) \right], \tag{1}$$

where $0 \leq \alpha \leq 1$ is the learning rate, r is the immediate reward received for executing option o in discrete state s (where $s \in \mathcal{S}_{ms}$), $0 \leq \gamma \leq 1$ is a discounted factor to indicate the preference between short term and long term rewards, and k is the duration of option o. In our task, $\alpha = 0.2$, $\gamma = 0.9$, and k takes the values shown in Table II.

D. Execution Phase

Once the policies for each motor system are stored in the robot's memory we can execute them to solve the task. However, the robot is now in charge of maintaining the visual memory, which does not contain any entity at the beginning. The robot has to look at entities in order to add its pair $(e_i, bel(e_i))$ into visual memory. Every time an entity e_i is observed, its belief $bel(e_i)$ (that represents its location in the low-level MDPs) is updated with new information.

1) Physical Action Selection: By having SMDPs built on a belief state, we now effectively have partially observable SMDPs (POSMDPs) [14]. The robot selects options for each of its motor systems by following the *Q-MDP algorithm* [9]:

$$o_{ms} = \arg \max_{o \in \mathcal{O}_{ms}} \sum_{s \in \mathcal{S}_{ms}} bel(s) Q_{ms}(s,o), \tag{2}$$

where o_{ms} represents the option with the highest expected reward for motor system ms, according to the belief $bel(s)$ of being in the discrete state s, and $Q_{ms}(s,o)$ is the Q-value taken from the policy π_{ms}. Because the motor systems are independent of each other in this task, the option selection can be executed in parallel with no problems.

To do the mapping between the high-level discrete state space (Table I), and the low-level continuous state of a particular entity location ($bel(e_i)$), we define the belief $bel(s)$ of being in a discrete state s as the probability distribution given by $bel(e_i)$. For instance, the discrete state variable *armPosition* takes the value *onObject* if and only if the offset of the hand's centre with respect to the object's centre is less or equal than some threshold (e.g. 1 cm). Each belief $bel(e_i)$ determines the likelihood of success or failure of options, which in turn determines changes in the discrete space.

The locations of entities are represented using particle filters, where each particle represents a possible location. The following equation redefines (2) in terms of particle filters:

$$o_{ms} = \arg \max_{o \in \mathcal{O}_{ms}} \frac{1}{|\mathcal{G}|} \sum_{g \in \mathcal{G}} weight(g) cost(g,o), \tag{3}$$

where $\mathcal{G}$ is the set of particles, and g refers to an individual particle. $weight(g)$ defines the weight given to particle g, and $cost(g,o)$ takes the value of $Q_{ms}(s,o)$ if the offset between the centre of the hand and the object is less than or equal to some threshold, otherwise it takes the value of $\min_{o \in \mathcal{O}_{ms}} Q_{ms}(s,o)$, indicating that the option failed.

2) Perceptual Coordination: The selection of good options depends heavily on having the correct state information. But only some information needs to be known to complete each step of the task, and the robot can choose which information to gather by pointing its cameras to a specific part of the world. However, since there is only one oculomotor system, it must be shared amongst all motor systems. The coordination mechanism works in the following way:

1) Each entity e_i listed in visual memory represents a fixation point, i.e. a perceptual action $p \in \mathcal{P}$, where $\mathcal{P}$ is the set of perceptual actions at any given time. The number of perceptual actions varies depending on the number of entities in visual memory.
2) We define ms_f as the motor system that will benefit the most if it is given access to perception:

$$ms_f = \arg \max_{ms \in MS} \{gain_{ms}\}, \tag{4}$$

where MS is the set of motor systems, and $gain_{ms}$ represents the expected gain that would result if gaze is allocated to motor system ms. The gain of each motor system is computed as:

$$gain_{ms} = \max_{p \in \mathcal{P}} \{V^p_{ms}\} - \max_{o \in O_{ms}} \{V^o_{ms}\}, \tag{5}$$

where p is a particular perceptual action, and $\mathcal{P}$ is the set of all possible perceptual actions. V^p_{ms} is the expected value for motor system ms assuming perceptual action p is taken. V^o_{ms} is the expected value with the current uncertainty. In fact, $\max_{o \in O_{ms}} \{V^o_{ms}\}$ has been calculated during the option selection using (2), so we can cache this value. The difference between the maximum of these two values tells us how much we gain if gaze is allocated to this motor system. To calculate V^p_{ms} we follow:

$$V^p_{ms} = \sum_{\omega \in \Omega} \left[P(\omega | bel, p) \max_{o \in O_{ms}} \sum_{s \in \mathcal{S}_{ms}} bel^{\omega}_p(s) Q_{ms}(s,o) \right] \tag{6}$$

where $P(\omega|bel,p)$ is the probability of making an observation ω given the current belief bel and perceptual action p. $bel^{\omega}_p(s)$ is the belief that results assuming we have taken perceptual action p and ω is the observation that results from this action. This equation is trying to predict the value of moving the gaze to a specific fixation point by using "imaginary" observations ω to update the current belief, and then checking the effect this possible new belief would have in the selection of options.

3) We define p_f as the perceptual action to be chosen once a motor system ms_f is selected using (4):

$$p_f = \arg \max_{p \in \mathcal{P}} \left\{ V^p_{ms_f} \right\}, \tag{7}$$

which is straightforward if we cache the results of $\max_{p\in\mathcal{P}}\{V^p_{ms}\}$ when calculating (5).

In our task, we just need to redefine (6) in order to handle particle filters:

$$V^p_{ms} = \frac{1}{|\Omega|}\sum_{\omega\in\Omega}\max_{o\in O_{ms}}\left(\frac{1}{|\mathcal{G}|}\sum_{g\in\mathcal{G}} weight(g,\omega)cost(g,o)\right), \tag{8}$$

where Ω is the set of observations, ω is a particular observation, $\mathcal{G}$ is the set of particles, and g is a single particle. $weight(g,\omega)$ represents the weight of particle g after having observed ω, and $cost(g,o)$ takes the value of $Q_{ms}(s,o)$ if the offset between the centre of the hand and the object is less than or equal to some threshold, otherwise it takes the value of $\min_{o\in\mathcal{O}_{ms}} Q_{ms}(s,o)$. Equation 8 updates a temporal belief $bel(e_i)$ of entity e_i which is not stored in visual memory since it is calculated from the imaginary observations ω.

These observations ω are sampled assuming that the robot would fixate on the mean of the cloud of particles ($mean(\mathcal{G})$)[1]. First, a number of particles g_i are uniformly selected. Then a bivariate Gaussian distribution is chosen for each g_i by indexing an observation model learnt off-line (described below), according to the location of particle g_i and the imaginary fixation point ($mean(\mathcal{G})$) with respect to the also imaginary oculomotor position. From the chosen bivariate Gaussian distributions the observations Ω are sampled.

3) Visual Analysis: When the robot's gaze fixates on some entity, visual input is read from the right camera. First, we detect entities appearing inside the camera's field of view (FoV). For detection we take advantage of the simulator; using the entities' true locations we calculate their 2D coordinates in the image plane[2]. The 3D location (in robot coordinates) of only those entities inside the FoV are added or updated in visual memory. A 3D location is calculated by triangulating the 2D coordinates of an entity with respect to the camera's position and the table's plane. Since only the right camera is used the triangulated locations are noisy[3].

In order to handle this noise during execution, an observation model was learnt off-line. This model is used for updating the particle filters and for sampling imaginary observations. The observation model was created by systematically moving the robot's gaze and an object in the robot space. Each fixation point and object location is represented using spherical coordinates: $(\theta_g, \phi_g, radius_g)$ and $(\theta_o, \phi_o, radius_o)$, respectively (Fig. 4 *A* and *B*). Whilst varying these parameters we recorded the offset between the estimated (triangulated) and the true object's location, resulting in more than 150,000 data points. These data points were divided into groups according to $(\theta_g, \phi_g, \theta_o, \phi_o, radius_o)$, then all the points of each group were fitted with a bivariate Gaussian distribution. There are 405 distributions in the final observation model, each representing a particular relationship between gaze and object location. Fig. 4 shows 3 of these distributions when the robot is fixating straight ahead, the circle in the centre of the graph represents an object. Distributions *C* and *E* represent the noise when objects are located at the left and the right of the fixation point respectively. Distribution *D* represents the noise when objects are located in line with the fixation point. Notice how the estimate in the location is much more accurate when the robot is looking directly at the object. Once the object starts to move away from the fixation point, in terms of $(\theta_o, \phi_o, radius_o)$, the estimate becomes very noisy. If an entity in visual memory is not seen for 3 seconds, Gaussian noise with zero mean and 1.0 cm of standard deviation, is added to its current estimate.

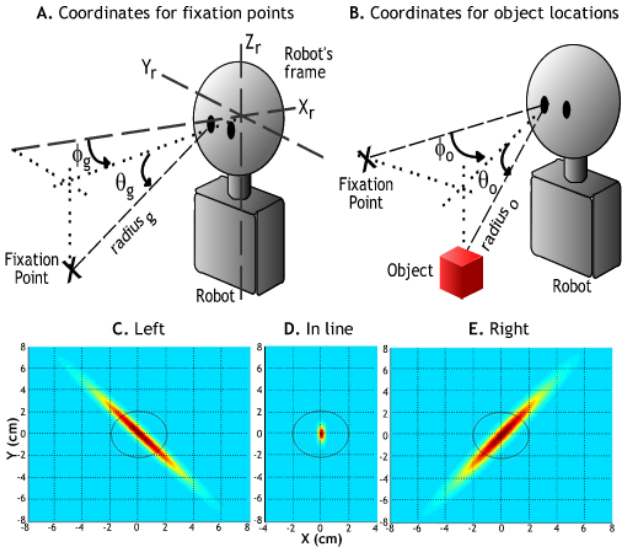

Fig. 4. Coordinates for the observation model, and distributions representing the noise during the triangulation of objects when the robot is looking straight ahead. A. The coordinates specifying fixation points. B. The coordinates specifying object locations. C. Objects appearing at the left of the fixation point. D. Objects in the fixation point. E. Objects appearing at the right of the fixation point.

III. Experiments

In order to test the effectiveness of our approach, we implemented two alternative gaze strategies: *random gaze allocation*, which randomly selects an entity from visual memory; and *round robin gaze allocation*, which loops through the current list of entities in visual memory.

As mentioned above, we can control the sensitivity of the options whilst reaching and grasping by varying a threshold. We conducted two sets of experiments, one for grasp sensitivity of 1 cm and another for 2 cm. For each gaze strategy we performed 15 trials of 5 minutes each. At the end of each trial we counted the number of objects correctly placed inside the containers. Fig. 5 compares the three strategies

[1]We could also fixate where the highest concentration of particles are, in the case of having multimodal distributions.

[2]By using the simulator for detection we end up with a noiseless object detection mechanism, which allow us to analyse our system just with the noise generated during triangulation. We can later add more sources of noise and analyse the effects that each one of them has in the system.

[3]Using both cameras for triangulation resulted in a perfect estimate, as the cameras are perfectly aligned in the simulator. Using one camera produces noisy estimates as it relies on a plane to calculate the depth.

in terms of the average number of objects correctly placed, with grasp sensitivity of 1 and 2 cm. The error bars in the graph represent 95% confidence intervals.

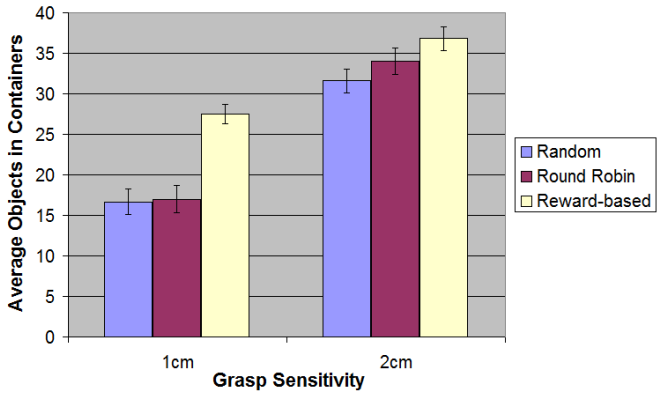

Fig. 5. Comparison between the three gaze allocation strategies in terms of the average number of objects correctly placed. The error bars represent 95% confidence intervals.

Using the two-tailed unpaired T-test, we verified that all the results are statistically significant at less than 0.019 level of significance. The results show that the reward-based strategy outperforms the other two strategies for both grasp sensitivity thresholds. The difference between the three strategies is clearer when the sensitivity is 1 cm. Here, the information about the location of entities needs to be more precise; some times it is necessary to fixate several times on the same entity in order finish some action, something that the reward-based strategy is able to handle but not random and round robin. When the sensitivity is 2 cm the performance of all strategies increases because fewer fixations need to be made for actions to be successful. Also, peripheral vision is important; any entity appearing inside the field of view will be detected and updated, as the sensitivity decreases it may not be necessary to fixate on some entities because the information gathered in the periphery might be enough to reach and grasp an object. This is mainly the reason why the difference in performance between the strategies is reduced. The decision time is an important issue as well. In average, the reward-based strategy takes 0.5 seconds or more (depending on the number of entities in visual memory), whilst random and round robin make a decision almost instantly. Even with this difference the reward-based strategy performs better.

Unfortunately, with our current model, it is not possible to compare our reward-based strategy against Sprague and Ballard's strategy. They model tasks with multiple concurrent goals that share a single motor system. Our current system models tasks with multiple motor systems, where each motor system executes a single goal. Once we extend our system so that each motor system is able to choose between multiple goals, it will be possible to directly compare both strategies.

IV. Conclusions and Future Work

The problem of how to coordinate gaze and actions in a robot is formulated in terms of a reward-based decision making framework, where camera movements are selected such that they reduce the uncertainty and increase the expected return on the task achieved by the motor systems. By controlling the direction of gaze, the robot should be able to decide which information is relevant at any point in time.

The results show that our reward-based gaze strategy outperforms a random and round robin strategies, whilst varying the sensitivity of the grasp actions to positional uncertainty. We identified two issues that should be taken into account in the analysis of the results. First, entities are detected and updated in the periphery of the field of view. This means that depending on the grasp sensitivity some entities might not be needed to be fixated at all, the information gathered in the periphery is enough. Second, the time to decide where to look. For round robin and random the decision time is almost instantly, whilst our reward-based strategy takes in average 0.5 seconds. This difference increases the performance of the non-informed strategies when the grasp sensitivity decreases.

Besides grasp sensitivity, we are investigating how our system behaves with different levels of observation noise, and also when the width of the field of view changes. Furthermore, we plan to extend the system to take into account concurrent motor systems. Another interesting extension would be to execute physical actions to help the perception process. For instance, if the object of interest is occluded by another object, it would be better to remove that object to get a better view.

References

[1] G. Metta *et al.*, "The iCub humanoid robot: An open platform for research in embodied cognition," in *Proc. ACM Performance Metrics for Int. Sys.*, MD, USA, Aug. 2008, pp. 50–56.

[2] L. Itti *et al.*, "A model of saliency-based visual attention for rapid scene analysis," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 20, no. 11, pp. 1254–1259, 1998.

[3] S. Frintrop, *VOCUS: A Visual Attention System for Object Detection and Goal-Directed Search*. New York, NY: Springer-Verlag, 2006.

[4] M. Sridharan *et al.*, "HiPPo: Hierarchical POMDPs for planning information processing and sensing actions on a robot," in *Proc. ICAPS*, Sydney, Australia, Sept. 2008, pp. 346–354.

[5] N. Sprague, D. Ballard, and A. Robinson, "Modeling embodied visual behaviors," *ACM Trans. Appl. Percept.*, vol. 4, 2007.

[6] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York, NY: Wiley-Interscience, 1994.

[7] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*. Cambridge, MA: MIT Press Cambridge, 1998.

[8] L. P. Kaelbling *et al.*, "Planning and acting in partially observable stochastic domains," *AI Journal*, vol. 101, no. 1-2, pp. 99–134, 1998.

[9] A. R. Cassandra, "Exact and approximate algorithms for partially observable markov decision processes," Ph.D. dissertation, Brown Univ., Rhode Island, May 1998.

[10] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA: MIT Press Cambridge, 2008.

[11] R. S. Sutton, D. Precup, and S. Singh, "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning," *AI Journal*, vol. 112, no. 1, pp. 181–211, 1999.

[12] U. Pattacini *et al.*, "An experimental evaluation of a novel minimum-jerk cartesian controller for humanoid robots," in *IEEE/RSJ Int. Conf. on Int. Robots, Sys.*, Taipei, Taiwan, Oct. 2010, pp. 1668–1674.

[13] S. Bradtke and M. Duff, "Reinforcement learning methods for continuous-time Markov decision problems," *Adv. in Neural Inf. Proc. Sys.*, vol. 8, pp. 393–400, 1995.

[14] S. Mahadevan, "Partially observable semi-Markov decision processes: Theory and applications in engineering and cognitive science," in *AAAI Symp. on Planning with POMDPs*, FL, USA, May 1998, pp. 113–120.