The acquisition of intentionally indexed and object centered affordance gradients : a biomimetic controller and mobile robotics benchmark.

Martí Sánchez-Fibla [‡], Armin Duff [‡] and Paul F.M.J. Verschure [‡,§]
{marti.sanchez,armin.duff,paul.verschure}@upf.edu

***Abstract*— We introduce affordance gradients (AGs), continuous sensorimotor structures that allow to predict the consequences of the agent's actions on the state of the environment. AGs allow to generalize among never performed actions and compress all possible consequences of the action state space. AGs also provide a way of estimating the world state after several interactions of the agent with objects. We validate the notion of AGs using benchmarks designed for mobile robotics that we solve using E-puck robot simulations: learn the affordances of several objects, push an object along a predefined trajectory and place an object in at a target position and orientation. We are interested in the neurophysiological basis of affordances and how they can be inserted in a sensorimotor loop with memory structures like the one proposed by the DAC architecture. We show that AGs provide a generalization of the perception-action couplets stored in memory and learned by they adaptive layer of DAC.**

I. INTRODUCTION

The learning of affordances has received a major interest from the robotics community in an scenario where an agent interacts and manipulates objects either using robotic arms, humanoid robots, [11], [14], [2], [12] and also mobile robots [13]. Affordances, as introduced by Gibson [1], are considered to represent all the action repertoires available to an agent in an environment. In [12], affordances are interpreted as the categorization of the change of state of an object after an action is executed on it. We follow a very similar approach here, but affordances can also be interpreted as learned correlations between predefined objects, actions and effects, as in [2]. Once acquired, affordances can then be used to predict world state configurations if an action would be performed, and thus are suited for object manipulation. In [12] for example, affordances are used for planning manipulations that require a sequencing of discrete actions. Object-Action Complexes (OACs) [3] introduce an additional level of logic description that can then be used to generate plans for manipulating more complex object configurations, like stacking boxes with a griper. Affordances in mobile robotics are also used to be able to act goal-directedly in unconstrained, dynamic environments [4], as in the case of assessing if an environment is traversable for a mobile robot [13].

Here, we focus on both, the process of acquiring internal representations of object affordances and how to use them for goal-directed manipulations. Our main contribution is the introduction of the notion of affordance gradients (AGs), object centered representations that describe the consequences that the agent's actions may have on a particular object. We call them gradients because they are represented as object centered force fields (in relation to the force fields used in the motor schemas based behaviors [7]). We call them intentionally indexed because an AG contains the information of the consequences of several actions. In this sense, AGs are goal-oriented because they allow us to interact with an object and predict what the consequences of our actions will produce on that object (in terms of physical movement) and allow to move it to a desired position with a desired rotation.

AGs are intended to generalize to actions that the agent has not previously performed and provide predictions to their consequences. AGs can then predict the position and orientation of an object after several interactions with it. If the agent does not have instantaneous, continuous access to a synchronization step of the allocentric world state (in this case the position and orientation of an object), this prediction will accumulate error when several actions are performed. It is conceptually the same error accumulation that odometry models have when the agent moves and does not have a global way of refining its assumed global position, for example using visual cues. Thus, as in the case of odometry where the agent can estimate changes of position over time, AGs can predict changes of the dynamic state of objects in the environment because of the consequences of the agent actions to the objects.

An affordance of an object is acquired through interaction and thus is a concept that cannot be understood in isolation, but it emerges from the combination of the continuous sensorimotor loop combined with memory. We have investigated affordances and the acquisition of sensorimotor contingencies [5] in the context of the Distributed Adaptive Control (DAC) [6], a biomimetic control architecture for artifacts. DAC is organized along three levels of control of increasing complexity: reactive, adaptive and contextual (see an adapted schema of DAC in figure 5). The reactive layer provides a pre-wired repertoire of reflexes, which enables the artifact to interact with its environment and exhibit basic behaviors. These basic behaviors provide cues for learning used by the adaptive layer. The adaptive layer acquires representations of sensory events and associated responses supporting the acquisition of simple tasks. The sensory and motor representations formed at the level of the adaptive layer provide the

Work supported by SF FP7-ICT-217148 and eSMC FP7-ICT- 270212.

[‡] SPECS, Technology Department, Universitat Pompeu Fabra, Carrer de Roc Boronat 138, 08018 Barcelona, Spain.

[§] ICREA, Institució Catalana de Recerca i Estudis Avançats, Passeig Lluís Companys 23, 08010 Barcelona

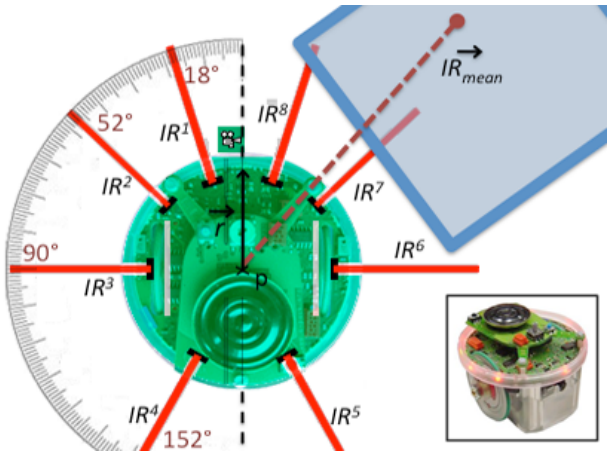

Fig. 1. Top view of the E-puck sensor model. A whole view of the E-puck is shown in the bottom-right corner. We indicate the position of the top/down camera and the angle in degrees of the 8 infrared sensors ($IR^{1\dots8}$). $IR^{5\dots8}$ have negative angles with respect to the front direction of the robot. For instance, $IR^8_\alpha = -18°$. The direction and position of the robot are denoted by $\vec{r}$ and p, respectively. We also plot a possible $\vec{IR}_{mean}$ computed from the IR reading when approaching a squared object.

inputs to the contextual layer, which acquires, retains, and expresses sequential representations containing perception-action couplets. These sensory motor pairs are the core representations used to express sequential behaviors. In a general sense, a sensory motor pair can be interpreted as a learned affordance as it associates a possible action to a sensory event. However, in this case the affordance is not related to a specific object but to possible actions of the artifact in the environment. In addition, the sequential organization of the sensory-motor pairs also allows predicting the consequences of the corresponding action as sensory events that come next in the stored sequence have a higher probability of occurrence. In [5] this concept has been extended to flexibly learn and relearn rules and to plan sequential behaviors to reach a predefined goal. The manipulation of objects has however not yet been regarded within the DAC architecture.

II. RELATED WORK

In what respects to the learning phase of an affordance, we follow a very similar approach than [11], [14], [12], that is, the learning is trained with a series of trials in which an action is executed and its effect/change in the object state is recorded (see section III-B). Our main contribution is that we propose an internal object-centred representation, the AG, that links action consequences (pushes in a precise location of an object in terms of displacement and rotation) with the object representation itself (defined in section III-A). This approach allows to approximate an intended direction of movement of an object when pushed from a particular location and a particular direction using a more compact representation than the target-motion direction maps and the roll probability maps of [11]. Very similar to [12], the effect of an action is the difference of the object state (called feature in [12]), before and after the action is performed, but in our case this effect is attached to a position of the shape of the object, the position from which it has been pushed. This makes the action space continuous for the same pushing direction. AGs are intended to generalize to actions that the agent has not previously performed (same idea in [14]).

Similarly to [12], AGs allow multistep planning and that is demonstrated with the task that we propose in which an object has to be placed in a target position and orientation. An AG allows to estimate the consequences of several pushes, an operation that accumulates error as explained in the context of algorithm 5, which implements a closed loop of iterative estimations and world reestimations until the goal state is reached (see section III-C). Another difference is that we use non symmetric objects like the tetris shape, and that the placement must be precise, so that a continuous action space is really needed to be able to perform the appropriate push according to displacement and rotation. In the case of [12], this precise positioning of non symmetric shapes would generate a complex plan of discrete actions. As we mentioned earlier, none of the tasks that we solve require contextual memory, thus, no search tree has to be expanded to find the best plan as is done in [12].

Another difference from [12] is that the control architecture that we propose is biologically plausible. AGs are seen inside DAC as generalizations of the action-perception couplets that the adaptive layer acquires. We deal with a previous stage of acquiring object-centred representations in [15]. In the latter, mainly inspired from rodent investigations, the discovery of affordances in an environment are a drive for exploration and part of a subsystem of a self-regulatory loop that seeks minimizing unpredictability of an environment.

III. METHODS

When learning affordances using mobile robots, usually because of their limited sensor models, we face the difficulties of estimating the world state with their local sensing capabilities. Here we use the E-puck robot [8] and mainly its infrared (IR) sensors, indicated in figure 1. We assume the robot has an odometry model that can estimate its position and direction in the environment: p and $\vec{r}$. Similarly to the allocentric robot position estimation provided by odometry, $\widetilde{\mathcal{O}} = \langle c, \vec{o} \rangle$ will denote the estimated position and orientation of an object, and thus reflects the estimation of the state of the external world. By inspecting the contour of an object and by averaging the IR readings we provide methods for estimating $\widetilde{\mathcal{O}}$. Using this estimation, the AGs of several objects are learned.

In section III-A we introduce the AG formulation and learning mechanisms. In III-C we use AGs for estimating $\widetilde{\mathcal{O}}$ after several interactions with an object. Our results have shown that AG allow for an integrated continuous sensorimotor decision gradient allowing to maintain an accurate prediction of the state of the objects in the world after several interactions with them.

A. AFFORDANCE GRADIENTS

From the action repertoires that an object can potentially serve (its affordances), we restrict ourselves to the "pushability" of an object, that is the consequences that the agent's actions have on the translation and rotation of the object. We define an AG of an object i as a tuple $\mathcal{A} = \langle S^i, P^i, R^i \rangle$. S^i is a 2D gradient, a two dimensional matrix, representing the shape of the object. S^i is centered at the center of gravity of the object $(0, 0)$ and arbitrary rotated in any direction. $S^i_{\langle x,y \rangle}$ is a value that indicates the probability, $0 \leq S^i_{\langle x,y \rangle} \leq 1$, that the object i has its limits/border at position $\langle x, y \rangle$. P^i is a 2D vector field where each point $P^i_{\langle x,y \rangle}$ is a vector representing the position where the center of the object will move when pushed form position $\langle x, y \rangle$ and direction $\overrightarrow{\langle -x, -y \rangle}$. R^i is a matrix where $R^i_{\langle x,y \rangle}$ indicates the angle rotation that the object after being pushed form position $\langle x, y \rangle$ and direction $\overrightarrow{\langle -x, -y \rangle}$. We also consider pushing directions deviated from the center direction of the object. We denote by $\mathcal{A}^\alpha$ an affordance learned when rotating the considered pushing direction $\overrightarrow{\langle -x, -y \rangle}$ by α degrees. This allows to learn affordances able to rotate the object with a small displacement. The three components of an AG allow us to estimate the new position and orientation of an object after a certain push. First we will describe how we can learn an AG and then how we can use it to push an object into a desired trajectory or how to place it in a desired position and orientation. Continuous affordance representations could then be abstracted to more general ones (pushable or not pushable, sliding or not sliding,...) also dependant from which side we approach the object (as done in [9] for a humanoid robot).

B. LEARNING OF AFFORDANCES

The $\mathcal{A}^\alpha$ affordance is computed by applying a push with a rotation of α degrees. From different push trials the robot fills the $\mathcal{A}^{0^\circ}$, $\mathcal{A}^\alpha$ and $\mathcal{A}^{-\alpha}$ structures. We proceed by explaining how we compute the components of the AG tuple $\mathcal{A}$. The robot approaches the object and stops when its about to make contact. It then pushes the object for a fixed duration setting a constant forward speed to the motors during a fixed period. A push can then be determined by a tuple $\mathcal{P} = \langle p_{obj}, \vec{d}, t_{millis}, s_{left}, s_{right} \rangle$, that is, the estimated contact point p_{obj}, the direction of push of the robot $\vec{d}$, a duration time t_{millis}, and the speed applied to the motors, s_{left} and s_{right}.

1) Learning of the shape gradient: For computing S, we add to the gradient evidence of presence of the limits of the shape in the direction of every IR sensor at each trial. For capturing with precision the irregularities, we estimate from each IR value, the distance to the border, and add the evidence at that location. One step of this process is shown in algorithm 1. We add a Gaussian to the shape gradient at the computed distance depending on the value of each of the four front IR sensors. This method is similar to the one used in [10] for computing the shape of an environment. There is no contact sensor in the E-puck, thus having contact with an object has to be inferred from the IR readings. The function $ObjInContact$ in the first line of algorithm 1, returns the estimated point of contact of the E-puck with the object and false otherwise. This point can be estimated in several ways. One can consider the mean of the points of contact estimated from the IR sensors as we explain below. One could use another sensor like a camera to determine it. We have tested experimentally that approaches like thresholding the IR sensors values are less accurate.

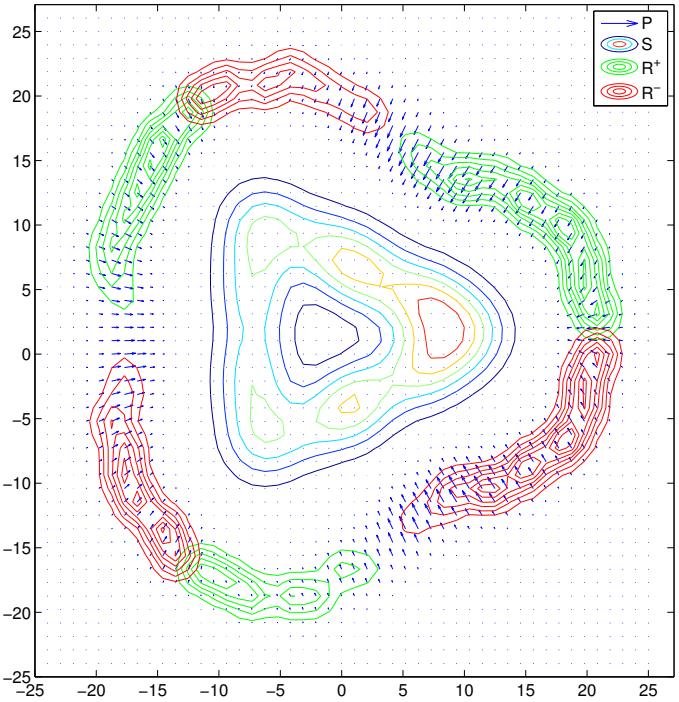

Fig. 2. Affordance gradient plot of a triangular object. We plot each component of the affordance: P the change of position vector field, S the shape gradient, R^+ the positive rotation gradient and the negative one R^-. This affordance corresponds to $\mathcal{A}^{0^\circ}$, when we approach the object facing its centroid. Other affordances, $\mathcal{A}^{-15^\circ}$ and $\mathcal{A}^{15^\circ}$, have not been plotted because of their similarity. For an online plot and usage of all the affordances see the video material supplied.

In figure 1 we show the E-puck sensor model. Red lines represent the range of the IR sensors. The IR sensors have a decay and we have assumed is quadratic. This decay has to be compensated to correctly estimate the position of the border of an object. This compensation is computed in line 2 of the algorithm and then multiplied by an arbitrary grid unit constant. We denote IR^i_{val} the value of the IR sensor i normalized to take values from 0 to 1, being the latter the closest that an object can be sensed. IR^i_α is the angle of the sensor with respect to the direction of the robot. In line 3 we compute the point of contact by summing the radius of the robot and the computed distance in the direction of the IR sensor with the agent position p. We then add a Gaussian in the S^i gradient shape at the computed point p_{obj} with sigma σ. Below IR values of 0.75 the reading is very noisy that's why we apply the restriction of line 1. In figure 2 we plot the learned shape gradient S.

Algorithm 1: ShapeGradientStep()

for $i = 1$ to 4 **do**
1 **if** $IR^i_{val} > 0.75$ **then**
2 $dist \leftarrow 30(\frac{1-IR^i_{val}}{0.25})^2 \vec{d}^{\angle IR^i_\alpha}$
3 $p_{obj} \leftarrow p + (radius + dist)\|\vec{d}\|$
4 $S \leftarrow S + g_{\sigma,p_{obj}}$

2) Learning of the position vector field: In algorithm 2 we depict an update step of the position vector field. The position of an object after a push is estimated from the IR sensors. To infer the center of mass of an object we combine the mean IR response, that is, the sum of the IR directions weighted by their IR value, with the direction of the robot (line 1 of the algorithm). In the formula, $\vec{r}$ represents the direction of the robot given by odometry. The operator $\angle$ used as super-index denotes the vector rotation operator. We show an example of the $\vec{IR}_{mean}$ vector in figure 1. When being in the learning phase, we assume that at each trial, the object is replaced at the same position and in the same orientation. Thus, we assume that the old position of the object p'_{obj} is in fact $\langle 0, 0\rangle$. The function $push$ just executes the predefined pushing of the object. The vector of the displacement is added to the vector field P by multiplying its components with a Gaussian centered at p'_{obj} (line 2). This is to smooth the final vector field so that we don't have under sampling problems.

Algorithm 2: PositionVectorFieldStep()

$p'_{obj} \leftarrow ObjInContact()$
$push(\mathcal{P})$
1 $\vec{IR}_{mean} = \sum_{i=1}^{8} IR^i_{val} \vec{r}^{\angle IR^i_\alpha}$
$p_{obj} \leftarrow p + \vec{IR}_{mean}$
2 $P \leftarrow P + g_{\sigma,p'_{obj}}(p_{obj} - p'_{obj})$

3) Learning of the rotation vector field: In algorithm 3 we estimate the orientation of an object in the environment, we call it the orientation vector of the object, and its position (its centroid). We do this by inspecting the contour of the object with the IR sensors and creating an estimated shape gradient $\widetilde{S}$, done by function $TurnArroundObjCompShape$. In line 1 we compute the centroid of the estimated gradient shape $\widetilde{S}$. In this case the function $computeCentroid$ does a straightforward sum of all the $\langle i, j\rangle$ coordinates in the gradient that are greater than a certain proportion of the maximum value. Lets suppose that these are n points. The centroid will then be in $\frac{\sum\langle i,j\rangle}{n}$.

After computing a centroid of this gradient (line 1), we center it at $\langle 0, 0\rangle$ (done by the function $Translate$) and we apply a best matching rotation procedure, keeping the best match at each iteration (line 3). The gradients are matched with its corresponding learned object shape gradient. This match is computed maximizing the number of non zero values after applying the AND operator (line 2). $TurnArroundObjCompShape$ returns the orientation vector $\vec{o}$ by rotating the unit vector using the angle of the best match.

We then add the corresponding angle to the rotation gradient in algorithm 4. This angle has to be added in the R^+ or R^- gradient depending on the sign of the displaced angle with respect to $\overrightarrow{\langle 1, 0\rangle}$), the default orientation of the object at each trial (see line 4). In line 5 the function $addMeanInRadius$ adds the mean of a previous angle value with the new one, or sets it for the first time if it didn't have a value yet. This is done in a certain radius to be able to overcome under sampling in the learning phase.

Algorithm 3: EstimateOrientationObj()

$\widetilde{S} \leftarrow TurnArroundObjCompShape()$
1 $c \leftarrow computeCentroid(\widetilde{S})$
$\widetilde{S} \leftarrow Translate(\widetilde{S}, c)$
for $\alpha = 0$ to 360 **do**
2 $s \leftarrow CountNonZero_{i,j}((\widetilde{S}^{\angle\alpha} \wedge S)_{\langle i,j\rangle})$
3 **if** $s > s_{bestmatch}$ **then**
$\alpha_{best} \leftarrow \alpha$
$s_{bestmatch} \leftarrow s$
return $\langle c, \overrightarrow{\langle 1, 0\rangle}^{\angle\alpha_{best}}\rangle$

Algorithm 4: RotationVectorFieldStep()

$p_{obj} \leftarrow ObjInContact()$
$push(\mathcal{P})$
$\langle c, \vec{o}\rangle \leftarrow EstimateOrientationObj()$
4 $a \leftarrow \angle(\vec{o}, \overrightarrow{\langle 1, 0\rangle})$
5 $R^{sign(a)} \leftarrow addMeanInRadius(p_{obj}, a, 10)$

C. USING AFFORDANCES

Algorithm 5 illustrates in very general terms, how AGs can be used to perform several interactions with an object, thus accumulating error in the estimation of its position and orientation $\widetilde{\mathcal{O}}$. If the number of iterations increases the agent can consider losing time performing a re-estimation of the object state. Inside this very fundamental loop one can also perform the tasks that require the usage of affordances: push along a line and positioning and orientation.

For using a learned affordance, the robot must be able to map it to an encountered object in the environment. For this purpose we estimate the orientation of the object with the procedure $EstimateOrientationObj$ described in previous algorithm 3. Consider that we could use a more general gradient matching procedure looking for the best match of all the learned objects in the memory. Once the AG has been mapped, the robot can plan the best push relative to the task it is performing (line 2). Once the push is executed, we can rely on the estimation provided by the $\mathcal{A}$ structure regarding its new position and orientation (line 3), done by the function $updateEstimate$, which has as inputs the estimation of the object position and orientation $\widetilde{\mathcal{O}}$ and the affordance $\mathcal{A}$, and returns the new updated estimation. We can then correct it using the actual value of $\vec{IR}_{mean}$. After

several iterations we may need to re-estimate the orientation of the object by a turn around exploration as the function $EstimateOrientationObj$ does (line 1).

The function $planBestPush$ can compute the best push angle giving priority to maximize the angle, the distance to the goal or both at the same time. We did a simple combined approach that is not exploiting the affordances at its maximum but it is sufficient to solve the task at a reasonable accuracy. While the distance to the final goal is greater than a threshold we search for the best push maximizing the pushed distance towards the goal. If this is not the case, we look at the difference in angle orientation. If the absolute value of the angle difference to the objective is more than 30° we look for maximizing the rotation, positive or negative depending on the sign. If it is less than 30° we look for a combined low rotation push and distance proportional to the remaining distance. This procedure is actually useful for both tasks taking into account that we don't care about the rotation angle when we want to push the object along a trajectory.

Algorithm 5: benchPusher()

$G \leftarrow buildGoalGradient()$
$findObj()$
while not success **do**
  if $acc < level$ **then**
1 $(\widetilde{\mathcal{O}} = \langle c, \vec{o} \rangle) \leftarrow EstimateOrientationObj()$
2 $\mathcal{P} \leftarrow planBestPush(\mathcal{A}, \widetilde{\mathcal{O}}, G)$
  $push(\mathcal{P})$
3 $\widetilde{\mathcal{O}} \leftarrow updateEstimate(\widetilde{\mathcal{O}}, \mathcal{A})$

D. AFFORDANCE GRADIENTS AND DAC

We want to emphasize with the adapted DAC schema of figure 5 the addition in the sensorimotor loop of the world state estimation step (in this case the state of objects) that occurs while also maintaining an estimation of the agent

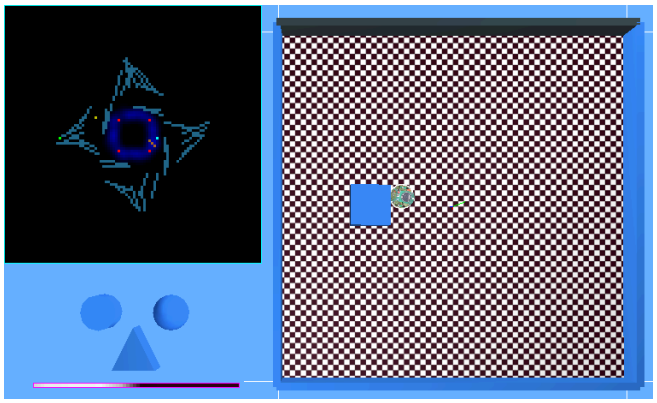

Fig. 3. Learning an affordance gradient of a cube. We show a snapshot of the first phase of the benchmark implemented in the Webots simulated environment. In the top left part we see a Webots display that represents the AG being learned by the robot. On the right we show a top view of the environment after a pushing trial of the E-puck robot. On the bottom left other objects are displayed.

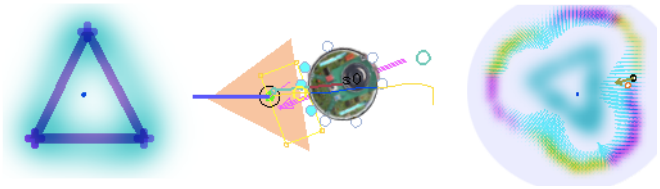

Fig. 4. The positioning and orientation benchmark. We show a snapshot of the second phase of the benchmark made with a custom simulation environment. The robot, in the center, must place the object at the goal gradient in the left. On the right we show the affordance computed in a previous phase (similar to the one explained in figure 2). The affordance is oriented at the estimation of the current orientation of the object.

position. This estimation is made by means of the AG structure which is a generalization of the action-perception couplets acquired by the adaptive layer. The learning of AGs needs a short term memory (STM) for remembering where the object was before the agent pushed it. This very particular use of memory is not typically contextual and can be achieved by sustained activity of a group of neurons.

We don't need the contextual layer for the tasks that we address here. Each task is solved by selecting the best push action with the instantaneous estimations of the agent position and world state. Contextual control would enter into play if one would need to interact with several objects and order of the manipulations would matter. It could also be necessary if the environment would contain fixed obstacles that would block objects in their way to the goal position.

The action selection module has access to the past internal states through the STM, can access the AGs and contains the goal gradient. It can thus select the action with the appropriate parameters from the different available motor programs. The module includes a timer for being able to execute fixed duration pushes.

IV. RESULTS

We validate the notion of AG using two benchmarks designed for mobile robotics. The first one consists in pushing objects along a predefined trajectory (line in the floor) and the second consists in placing an object at a goal position and orientation. Both tasks are preceded by a training phase where the aim is to learn the AGs.

All tasks are solved in simulations. We have implemented a custom simulation environment and also used Webots, Cyberbotics Ltd. If the AGs are learnt from the IR sensors of the E-puck robot, one can assume a flat world without loss of generality. For the sake of simplicity we implemented a 2D simulator (we show a snapshot of the simulator while the robot is performing the positioning and orientation task in figure 4). For dealing with the 2D physics we have used Box2D library.

We present in the following the results of the learning phase. For computing each AG $\mathcal{A}^\alpha$, we execute 60 trials corresponding to a whole turn around of the object with steps of 6° degrees. $\mathcal{A}^{0^\circ}$ is shown in figure 2. As expected, the position vector field is higher in the center of the three faces, showing the ability to displace the object maximally when

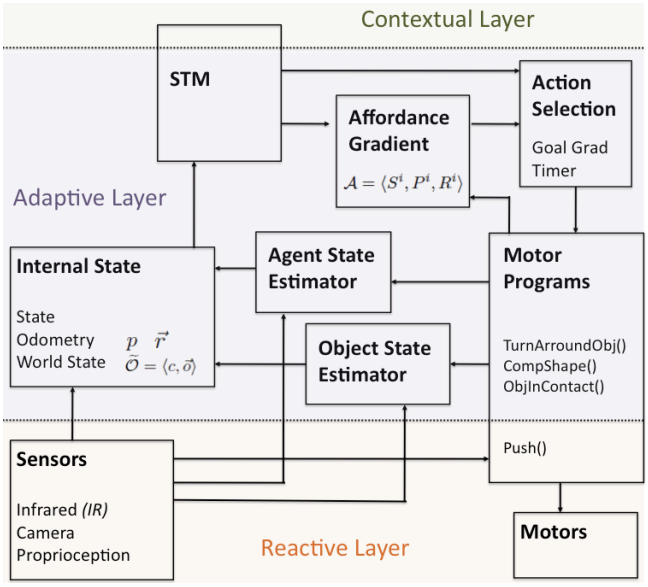

Fig. 5. DAC schema. The adaptive layer contains the main ingredients for the world state and agent position estimation. See text for further explanation.

pushing from this position. The rotation angle at this point is very low, which is consistent with the fact that the object maintains its orientation when pushed. The rotations R^+ and R^- are maximum when a triangular shape is pushed around the corners. Using $\mathcal{A}^{15^\circ}$ and $\mathcal{A}^{-15^\circ}$ bigger rotations can be achieved. For $\mathcal{A}^{0^\circ}$, $\mathcal{A}^{15^\circ}$ and $\mathcal{A}^{-15^\circ}$ we get a maximum rotation (postive or negative) in a single push of 39.51°, 59.06° and 79.78° respectively.

We also implemented a first version of the AG learning using Webots, we show a snapshot in figure 3. This was to be able to deal with objects that cannot be simplified into 2D representations, like a ramp. In this case AGs allow the agent to represent, for example, that a ramp is pushable from one side and can be rolled from the other and this being a previous stage before being able to manipulate the object.

We present now the results of the pushing along a trajectory task. For pushing objects that have several symmetry axis along a predefined trajectory its is actually not necessary to learn their affordances. It can suffice to estimate its centroid and push the object in the desired direction will most probably move the object in that direction. We prove nevertheless that using a learned affordance can lead to a more precise pushing trajectory. In figure 6 we show two different runs of the pushing along a trajectory benchmark: in the left using the computed affordance and in the right without using it. As we mentioned, it is important for this benchmark that the piece has no axis if we really want to exploit AGs. This is the reason why we selected the l-shaped piece. To quantify precisely which trajectory is better we introduce the following measure. We sum for every point the distance to the closest point of the predefined trajectory, we then divide the sum by the number of points. Using affordances we get an error of 7.82 units per point and

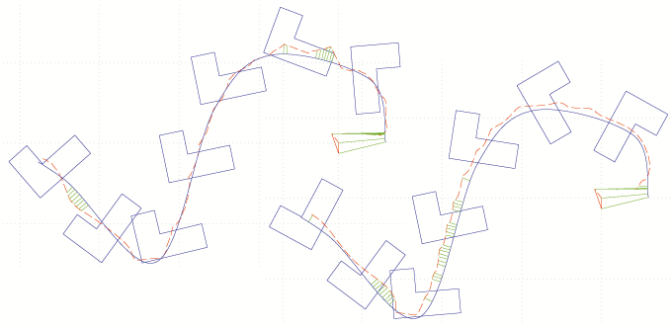

Fig. 6. Pushing objects following a trajectory. Left trajectory uses the learned affordance and the right one does not. We highlight the normal vectors in green when the distance from the centroid of the object and the trajectory exceeds 10 units. The object contour is plotted every 30 steps.

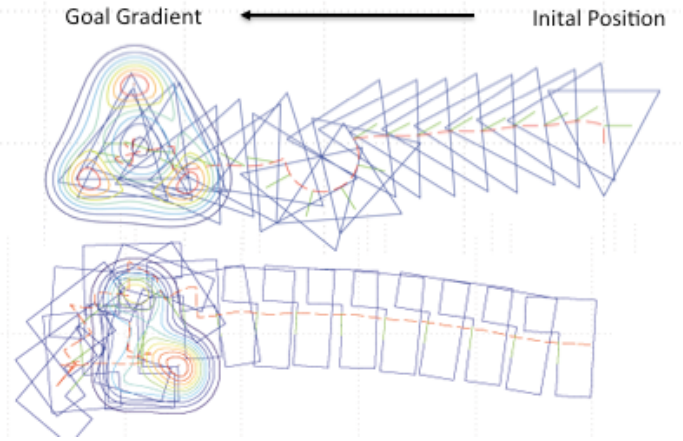

Fig. 7. Positioning and orienting an object. The top row corresponds to a run pushing the triangular piece and the bottom row using the l-shaped piece. The goal gradient where the object has to be placed is superimposed in the left.

without using them we get 10.71.

We present now the results of the position and orientation benchmark. This task proves the real need of affordances and cannot be performed without using them. To place an object in a precise target orientation, the agent needs an internal representation of the object and knowledge of how the pushing actions affect the rotation of the object. This is not the case of the circular object which is symmetrical in all axes, so does not need to be rotated. In figure 7 we show two runs of the positioning and orientation benchmark, using the triangular and l-shaped pieces. We plot the position of the shape every 5 time steps when being pushed towards the goal gradient. In the triangular piece run it can be observed at the middle of the route that algorithm 5 switched to adjust the rotation angle and then continued approaching the goal. In the l-shape run this switch occurs when the piece is closer to the goal position.

V. CONCLUSIONS

We have introduced the notion of affordance gradients (AGs), continuous discrete surfaces for adaptive behavior. We have set the theoretical framework for learning AGs

using a mobile robot. We believe that little work is needed to generalize the successful simulations that we performed using the E-puck robot to a real mobile robot.

Coming from the simple affordances utilized in the DAC architecture we have generalized the concept of action-perception pairs to AGs which can be used for object manipulation. AGs improve DAC's acquired affordances in multiple ways. First, AGs are object centered and compress the different possible actions in a single representation. In DAC each possible encountered action is stored in a separate memory element also leading to overrepresentation of the same perception action. Secondly the explicit representation of the effect allows for a straightforward implementation of goal oriented action. In DAC the effect of an action is only represented implicitly in the sequential organization of the action-perception pairs and has to be retrieved by action propagation through the network [5]. Thirdly AGs allow to continuously interpolate between different actions leading to a more general representation. DAC has been successfully applied to explain rodent foraging behavior [10]. The AG allow to extend this approach towards object manipulation. Little is know about object perception and affordance acquisition in rodents (see [16] for a recent review). In this context, affordances are considered as relations between the abilities of animals and the properties of objects. Experiments show that object properties are accessible along various dimensions such as shapes, textures, odour, color and brightness. Although this vast range of recognition characteristics, rats show a preference for objects that have affordances for common rat activities, for example, in [17] it is found that rats prefer objects they could climb onto, to those they could not. Rats also show interests for the manipulation (grasping,pushing,...) of objects that could interfere with accessing new unexplored areas of the arena, alleys, corridors, and thus could be rewarding in a later state. In the same line, it seems also that novelty of an object is an important aspect, increasing the interest of the rat in exploration. Using these ideas we designed a robotic experiment in [15] in which an agent explores an environment with the aim of minimizing its predictability which also includes the affordances available in the environment. In [15] nevertheless, affordances are not represented internally by the agent. This was the purpose of this paper by introducing the AGs.

The presented benchmarks have also been informed by these rodent investigations, but to be able to reach a level of interest for the robotics community we used a task rats probably do not perform. In mobile robotics the usage of affordances is rarely considered. [13] makes use of the traversability affordance to navigate through an environment but it lacks of a more precise use of the learned structures.

The generality of our approach would allow for example to modify the morphology of the robot attaching to it an additional part and then be able to relearn the new affordances related to the action consequences of having this new body part. We are currently investigating this generalization for tool use and body morphing.

VI. ACKNOWLEDGMENTS

This work was supported by Synthetic Forager FP7-ICT-217148 and eSMC FP7-ICT- 270212. We would like to thank partners in the eSMC consortium for helpful discussions.

REFERENCES

[1] J. Gibson, *The ecological approach to visual perception*. Lawrence Erlbaum, 1986.

[2] L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor, "Learning Object Affordances: From Sensory–Motor Coordination to Imitation," *Robotics, IEEE Transactions on*, vol. 24, no. 1, pp. 15–26, 2008.

[3] C. Geib, K. Mourao, R. Petrick, N. Pugeault, M. Steedman, N. Krueger, and F. Wörgötter, "Object action complexes as an interface for planning and robot control," in *IEEE RAS International Conference on Humanoid Robots*, 2006.

[4] E. Rome, L. Paletta, E. Sahin, G. Dorffner, J. Hertzberg, R. Breithaupt, G. Fritz, J. Irran, F. Kintzler, C. Lörken *et al.*, "The MACS project: an approach to affordance-inspired robot control," in *Proceedings of the 2006 international conference on Towards affordance-based robot control*. Springer-Verlag, 2006, pp. 173–210.

[5] A. Duff, M. Fibla, and P. Verschure, "A biologically based model for the integration of sensory-motor contingencies in rules and plans: A prefrontal cortex based extension of the Distributed Adaptive Control architecture," *Brain Research Bulletin*, 2010.

[6] P. F. Verschure, T. Voegtlin, and R. J. Douglas, "Environmentally mediated synergy between perception and behaviour in mobile robots," *Nature*, vol. 425, no. 6958, pp. 620–4, 2003.

[7] Arkin, R. C., *Behavior-Based Robotics*. MIT Press, 1998.

[8] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J. Zufferey, D. Floreano, and A. Martinoli, "The e-puck, a robot designed for education in engineering," in *Proceedings of the 9th conference on autonomous robot systems and competitions*, vol. 1, no. 1, 2009, pp. 59–65.

[9] B. Akgun, N. Dag, T. Bilal, I. Atil, and E. Sahin, "Unsupervised learning of affordance relations on a humanoid robot," in *Computer and Information Sciences, 2009. ISCIS 2009. 24th International Symposium on*. IEEE, pp. 254–259.

[10] M. Sanchez-Fibla, U. Bernardet, and P. Verschure, "Allostatic control for robot behaviour regulation: An extension to path planning," in *I. Robots and Systems, IROS 2010*. IEEE/RSJ, pp. 1935–1942.

[11] P. Fitzpatrick, G. Metta, L. Natale, S. Rao, and G. Sandini, "Learning about objects through action-initial steps towards artificial cognition," in *Robotics and Automation, 2003. Proceedings. ICRA'03*, vol. 3. IEEE, 2003, pp. 3140–3145.

[12] E. Ugur, E. Oztop, and E. Sahin, "Goal emulation and planning in perceptual space using learned affordances," *Robotics and Autonomous Systems*, 2011.

[13] E. Ugur, M. Dogar, M. Cakmak, and E. Sahin, "The learning and use of traversability affordance using range images on a mobile robot," in *Robotics and Automation, 2007*. IEEE, 2007, pp. 1721–1726.

[14] A. Stoytchev, "Learning the affordances of tools using a behavior-grounded approach," *Towards Affordance-Based Robot Control*, pp. 140–158, 2008.

[15] M. Sanchez-Fibla, A. Duff, U. Bernardet, and P. Verschure, "A biomimetic robot controller based on minimizing the unpredictability of the environment: allostatic control revised," in *European Conference on Artificial Life, ECAL*. MIT Press, 2011, In Press.

[16] A. Ennaceur, "One-trial object recognition in rats and mice: Methodological and theoretical issues," *Behavioural Brain Research*, 2010.

[17] A. Chemero and C. Heyser, "Object exploration and a problem with reductionism," *Synthese*, vol. 147, no. 3, pp. 403–423, 2005.