CHIME 2

Monday, 19 October 2020 23:58

- Hierarchical affordance discovery using intrinsic motivation:
- Affordance learning: learn the relation between observed physical properties in the environment to possible interaction.
- Mechanisms learnt from human life-long learning.
  1. the way infants relate what they see to how they may interact with surrounding objects.
     The concept of affordance [Gibson'79] [whitehead'81] consists of ① embodiment and ② motor capabilities.
  2. how the infants explore and interact with thier environment while building new skills. (curiosity)
     The capacity is described as intrinsic motivation [Miller'88] which provides a powerful mechanism to learn motor skills such as affordances.
- Staged development of robot skills [Ugur'15]
  propose a developmental approach of affordance learning; the robot learns by stages; simple affordances → complex
  but they limit the approch to simple affordances, with no multi-object interaction possible.
- Methods using intrinsic motivation to build a hierarchy of interrelated skills:
  1. Overlapping waves in tool use development [Forestier'16]
     (pre-programmed and static hierarchies).
  2. Learning a set of interrelated tasks by using a succession ... [Duminy'19]
     (learning through exploration).
- CURIOUS [Colas'19]:
  proposed combining DRL and intrinsic motivation to generate goals to explore.
- CHIME couldn't generalize its skills to new objects.
- Problem formalization:
  - A robot interacting with its non-rewarding environment by performing sequences of motions of unbounded length in order to induce changes in its surroundings.
  - Sequences of motions are primitive actions described by a parametrized function with N parameters $a \in A \subset \mathbb{R}^N$,
  - Primitive action (a): corresponds to a command that me be sent to one or several actuators of the robot.
  - Sequence of primitive actions $a = [a_1, \dots, a_n] \in A^N$
  - Changes are the consequences on the environment, observable by the robot is called observations $\omega \in \Omega \subset \mathbb{R}^M$
- Formalization of the approach:
  - The robot learns models of relations $a \to \omega$ within a given context $\tilde{\omega}$ (absolute state before executing the action).
  - The controllable ensemble: $C = A \cup \Omega_{controllable}$
    is a model may be used to find one or a sequence of primitive actions to be performed in order to induce a value for the given observable.
  - Affordance model $A(C_i, \Omega_j, \tilde{\Omega}_k)$
  - To visually identify affordance, we associate A with a visual predicator P_A; it learns Ω and indicates whether A maybe be applied to an object o in the scene or not.
  - A possesses ① a forward model (M_A) and ② an inverse model (L_A).
  - Each affordance A can be seen as a basic skill.
  - $\mathcal{H}$ is the ensemble of the affordances used by a robot, and varies along time.
- Global architecture:

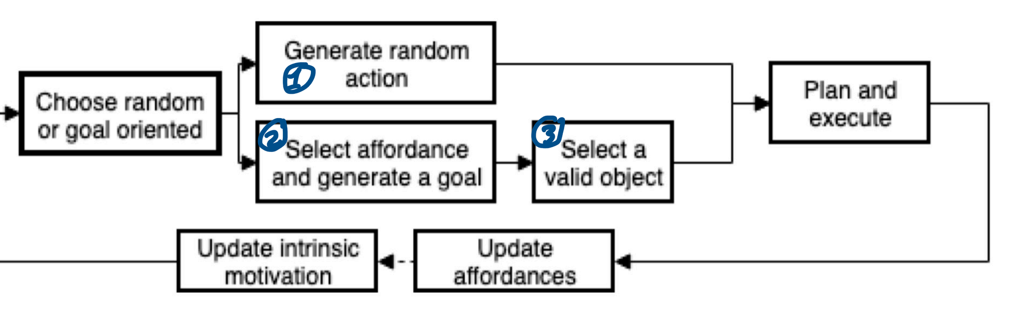

Figure 2: Abstract layout of a learning episode, beginning is on the left on the bold node.

- at each iteration, one primitive shape is performed.
  ① the robot generates a random controllable $c \in C$, and if required convert it to an executable primitive action.
  ② an affordance A and a goal ω_g are selected based on an intrinsic metric → ③ the robot selects the closest object o considered valid by the affordance visual classifier P_A
  → the robot uses its inverse models and its planning system to infer a sequence of controllables c to reach ω_g
  - After each episode, the robot obtains a list $(a^i, \omega_1^i, \dots, \omega_k^i, \tilde{\omega}_1^i, \dots, \tilde{\omega}_k^i)$
    i - iteration index, k - number of subspaces of Ω.
  - Data are then ① stored in D
    after processing ② used to improve existing affordances.
    ③ decide whether creating a new affordance is necessary or not.
    ④ update the intrinsic motivation system.
- Algorithm:

Algorithm 1 Algorithm layout

```
i = 0
loop
    D_episodic = ∅                                      Random
    if H ≠ ∅ and [Random() ≤ σ] then
        A = AffordanceSelection(H)
        ω = GoalSelection(A)
        ω_g = ObjectSelection(A, ω)
        c = Plan(ω_g)
    else
        C_i = RandomControllableSpace(C)
        c_r = RandomValue(C_i)
        c = [c_r]
    a = TransformToPrimitive(c)
    for a_k ∈ a do
        ω_before = GetObservations(Ω)
        a_i = [c_k] if c_k ∈ 𝒜 else TransformToPrimitive(c_k)
        Execute(a_i)
        ω_after = GetObservations(Ω)
        ω_i = ω_after − ω_before          ← Δω
        ω̃_i = ω_before
        D_episode ← (a_i, ω_i, ω̃_i)
        i += 1
    UpdateInterestMaps(D, D_episodic)
    UpdateAffordances(D, D_episodic)
    D ← D_episodic
```

- Intrinsic motivation:
  SAGG-RIAC R-IAC [Baranes'09]
  - for each affordance A the system creates an interest map; a partition of Ω_A that is constructed incremently based on progress measures;
  - the goal is to divide Ω_A into regions and attribute a value of interest to each region.
  - the interest corresponds to a monitioring of how much exploring this region may improve the robot knowledge in the future.
  - the competence of an affordance A near a goal ω is the mean $(\omega_e - \omega_r)$ for the last k last outcomes near ω
    ω_e - an outcome goal estimated by the algorithm for a given controllable c.
    ω_r - the effective outcome reached during exploration.
  - learning progress is the derivative of the competence.
  - the interest value of a region corresponds to the mean of the last n learning progresses in the region.
- Actions and controllable execution:
  if $c_i \notin A$ but $c_i \in \Omega_{controllable}$ then an affordance A is selected with $c_i \in \Omega_A$ and its inverse model is applied onto c_i to obtain low level controllable $b_i \in C_A$
  or a planning phase is used to plan a sequence of elements of C_A in order to reach c_i when executed.
- Affordance addition and update:

Algorithm 2 Autonomous affordances adaptation

Input: a the actions performed during the episode, ω the observations at the beginning of each iteration of the episode.

```
Spaces = SelectSpaces(Ω, ω)
repeat k times
    S = PickSpace(Spaces) → randomly
    for A ∈ H do
        matched = False   ①
        if Matches(A, a, ω_S) then
            matched = True
            Add (a, ω_S) to the model training dataset of A
            TrainVisualClassifier(A, ω_S, True)
        else
            repeat k' times
                S'_context = PickSpace(Ω)   ← try to add context spaces
                NewA = Copy(A)                to the affordance
                ContextSpace_NewA = ContextSpace_NewA ∪ S'_context
                if Competence(NewA) ≥ τ_modification then
                    A ← NewA
                    matched = True
                    break
    p_A ← TrainVisualClassifier(A, ω_S, matched)
    if matched then
        Add a, ω_S to the model training dataset of A
    else
        NewA = Affordance(a, S, ∅)
        if Competence(NewA) ≥ τ_creation then    } tries to create new affordances
            H ← NewA
            p_NewA ← TrainVisualClassifier(NewA, ω_S, True)
```

① A space matches an affordance if adding the data from this space to the training set of affordance doesn't reduce its competence.

- The algorithm requires 2100 iterations to reach the $\underbrace{\text{final level of competence}}_{0.96}$ of $\underbrace{\text{CURIOUS}}_{32000 \text{ iterations}}$.