Plan-based Control in an Affordance-based Robot Control Architecture

Joachim Hertzberg, Christopher Lörken
Institute for Informatics

www.inf.uos.de/kbs/

Thanks to
Andreas Bartel, Kai Lingemann, Frank Meyer, Andreas Nüchter, Stefan Stiene
Planning & Execution in MACS
From the WP4 Objectives

• Understand interplay *sensor data* – *action usage* – *affordances* in robotics

• Understand inter-relation *affordances* – *symbol grounding* in robotics

• Specify generic representations & data structures supporting affordance usage

• Understand how affordances reified as data structures interact with signal-to-symbol conversions, action structures, and qualitative representations of objects

• Propose software module supporting a *sensor data* – *dynamic object* – *action execution* – *sensor usage* loop on a robot

• Study the relation *affordances* – *planning for sensor use*, where affordances may be viewed as constraints on sensor use

• Propose sensor planner operating dependently with the above
UOS’s Contribution to WP4 Objectives

Use s-o-a AI planning technology (PDDL language, FF planner)

Top-down influence on affordance usage
- Focus attention on affordances serving current operator; disregard others
- Search actively for cues signaling focused affordances

Plan-Based Robot Control

Affordance-Based Robot Control

Ground (some) domain concepts in \(\langle c, b, o \rangle_s\)
- Tailor (some) operators after afforded actions
- Use affordances as preconditions for “opportunistic” execution
- Map perceived affordances
- Don’t distinguish among functionally equal objects
- Reduce search

Use the MACS modules exec. control, behaviors, affordance repository

Affordance-Based Robot Control

Plan-Based Robot Control
Background: (Propositional) AI Planning

• Dates back to STRIPS/SHAKEY tradition in AI

• Modern algorithms ("neo-classical planning") by orders of magnitude faster, termination guaranteed
  • We re-used FF (Hoffmann/Nebel, 2000’s)

• Well-understood on formal grounds
  • Situation is a set of ground facts
  • Operator pre/postconditions sets of ground literals
  • Plan is a partially ordered operator set
  • Decidable, NP-hard wrt. domain size

• De-facto standard domain descr. language PDDL (+ variants)
PDDL: s-o-a Domain Descriptions

• Originally developed to facilitate the International Planning Competition (IPC)

• Codifies input syntax for specifying planning domains & problems in terms of
  • predicates (proposition schemata)
  • actions
  • objects
  • start situation, goal propositions
  • optional requirements (typing, equality handling, …)

• Various upgrades exist for enhancing expressivity (time, …)
Background: Robot Planning

The plan is that part of the robot’s program, whose future execution the robot reasons about explicitly.

[D. McDermott, 1992]

- Dates back to STRIPS/SHAKEY tradition in AI
- Various benefits for robot ctrl: Performance optimization (time, robustness), communication, software engineering
- Plan is just one source of information for robot control (hybrid control architectures)
  - “Sense-Model-Plan-Act” (SMPA) loop is a straw man!
- Plan format may vary; notion of planning may differ from classical view (“adapting library plan stubs”)
  ➤ Autonomous execution matters
  ➤ Needs symbol grounding/object anchoring & action grounding
Example: MACS Arena Affordance Map

Topological Region (fuzzy boundaries)

Perceived Affordance (arbitrarily often; possibly different cues)

While being in some environment, log perceived affordance types per region! (1 entry / type / region)
MACS DD Example: Predicates

(:types region switchRegion doorRegion room)

(:predicates
  (robotAt ?region - region)
  (inRoom ?region - region ?room - room))

(hasLiftedSomething)

(liftable ?region - region)

(switch-triggerable ?region - switchRegion)

(passable ?startRegion ?targetRegion - region)

Model affordances by properties of the regions where they have been perceived
(no objects sneaking in through the backdoor)
MACS DD Example: An Operator

(:action lift
 :parameters (?region - region)
 :precondition
 (and
 (robotAt ?region)
 (liftable ?region)
 (not (hasLiftedSomething)))
 :effect
 (and
 (hasLiftedSomething)
 (not (liftable ?region))))

Grounded by localization
Grounded by affordance (map)
Grounded by “introspection”

Side effect:
delete liftability tag for
?region in aff. map
Execution: Grounding Operators

• Ground operators in behaviors (hybrid architecture)
  • e.g., lift operator is implemented using:
    DirectGoToPoseBehavior, 3DScanBehavior, ReachBehavior, PullBehavior, RaiseBehavior

• Specialty induced by affordances:
  If an operator corresponds to an afforded action,
  then grounding is provided by the $b,o$ in $\langle c,b,o \rangle$!

• Execution monitoring of afforded action means to “go with the flow of affordance”
  (But only the selected one!)
Example: Grounding the Lift Operator

Grounding the Lift Operator

Christopher Lörken, Frank Meyer, Joachim Hertzberg

Knowledge-Based Systems Research Group

© Lörken, UOS-KBS 2008
Opportunistic Execution

- At operator execution, perception is primed to attend to cues relevant for current operator execution
- Any(!) entity affording what is needed may be used (“lift something” vs “lift object O_17”)
- Purely object-based representations handle poppycock (“get me Glass_42 with water” instead of “get me a glass of water”)

☛ Using entities&affordances in addition to(!) objects&properties appears to make a lot of sense!
(define (problem macs-prob) 
  (:domain macs-example) 
  (:objects rightRoom - room 
    leftRoom - switchRoom 
    region1_left region2_left region1_right - region 
    switchRegion - switchRegion 
    doorRegionLeft doorRegionRight - doorRegion) 
  (:init (inRoom region1_left leftRoom) 
    (inRoom region2_left leftRoom) 
    (inRoom switchRegion leftRoom) 
    (inRoom doorRegionLeft leftRoom) 
    (inRoom region1_right rightRoom) 
    (inRoom doorRegionRight rightRoom) 
    (robotAt region1_left) 
    (liftable region1_left) 
    (switch-triggerable switchRegion)) 
  (:goal (robotAt doorregionright)))
Complete Example (Simulator)

The Goal

(:goal (robotAt doorregionright)))

The Plan (FF generated)

0: LIFT region1_left
1: CARRY region1_left switchregion
2: TRIGGER-SWITCH switchregion
3: APPROACH-REGION switchregion doorregionleft
4: CHANGE-ROOM doorregionleft doorregionright
Action Planning in the MACS Robot Control Architecture

Christopher Lörken, Frank Meyer, Joachim Hertzberg

Knowledge-Based Systems Research Group

© Lörken, UOS-KBS 2008

www.inf.uos.de/kbs/
Execution Failure

• Planned operator execution by afforded actions may fail due to
  • Model error (affordance not present where in map)
  • Perception error (cue is there but gets overlooked; affordance is perceived false-positively)
  • Handling error (afforded behavior fails)

• Reaction inventory in plan-based control: retry, replan, give up

• Afforded actions may be retried by using different affordance instance of the same type (perceived or looked up in map)
Execution Failure etc., Simulator Expl.

Affordance-based Planning with Erroneous World Models

Christopher Lörken, Frank Meyer, Joachim Hertzberg

Knowledge-Based Systems Research Group

© Lörken, UOS-KBS 2008
**Future Work**

Examine interplay plan-based & affordance-based control

- Continue/extend experiments (real robot, opportunism, execution failure)
- Examine more expressive plan language (time)
- Interface with learning
- Integrate individual objects
Summary of Contributions

The interplay between affordance-based and plan-based robot control has been explored for the first time

- p-b ctrl helps a-b ctrl focus top-down on relevant affordances
- a-b ctrl helps p-b ctrl ground actions and predicates
- employed s-o-a propositional planner in robot ctrl
- found operational model for opportunistic plan execution
- found simple mechanism for handling anonymous entities/objects in propositional robot planning
... and integrated it all in the MACS architecture
Publications

… during MACS funding 2007 and after; beyond deliverables


