IROS 2012 Workshop (SW5)

Cognitive Assistive Systems (CAS 2012): Closing the Action-Perception Loop

7 October 2012
Vilamoura, Algarve, Portugal
Room: Lince

Contents

Motivation **2**

Organisers **3**

Reviewers **4**

Invited Talks **5**

1 Prof. Yiannis Aloimonos
Cognitive Robots with a minimalist action grammar: Theory and Applications 5

2 Prof. Danica Kragic
Extracting and representing relevant information from high-dimensional data 5

3 Dr. Jeremy Wyatt
Active sensing and prediction in cognitive robots 6

4 Prof. Tamim Asfour
Combining Active Vision and Active Touch for Grasping Unknown Objects 6

5 Dr. Claude Andriot
Immersive Virtual Manufacturing and Training with Haptic Feedback and Virtual Manikins . 6

Accepted Papers **8**

1 Multi-scale cortical keypoints for realtime hand tracking and gesture recognition
M. Farrajota, M. Saleiro, K. Terzic, J.M.F. Rodrigues and J.M.H. du Buf
University of the Algarve, Portugal 9

2 Active Perception of Objects for Robot Grasping
J. Bimbo, X. Song, H. Liu, L. Senerivatne, and K. Althoefer
Kings College London, United Kingdom 15

3 A Preliminary Account of 3D Visual Search
F. Pirri, M. Pizzoli and A. Sinha
Sapienza University of Rome, Italy 19

4 Symbiotic-Autonomous Service Robots for User-Requested Tasks in a Multi-Floor Building
M. Veloso , J. Biswas , B. Coltin , S. Rosenthal, S. Brandao, T. Mericli, and R. Ventura
Carnegie Mellon University, USA
Superior Tecnico, Portugal
Bogazici University, Turkey 25

5 Decoupling Behavior Control and Perception in Affordance-Based Manipulation
T. Hermans, J. Rehg, and A. Bobick
Georgia Institute of Technology, USA 31

6 An in-field magnetometer calibration method for IMUs
S. Salehi, N. Mostofi, and G. Bleser
German Research Center for Artificial Intelligence, Germany 39

Motivation

It is becoming increasingly clear that future robotic systems will need to exhibit sophisticated assistive capabilities, highly tuned and responsive to the needs of human users. Whether on autonomous platforms or within personal computing systems, awareness of human intentions and requirements will be an essential attribute of any robotic system aiming to be genuinely useful. In essence, they will need to be capable of empathising with human behaviour if they are to be truly assistive in the fullest sense of the word.

Realising such cognitive assistive (CA) systems will require advances along the complete processing pipeline, from sensing through to learning and interaction. For instance, sensing will need to be proactive, anticipating user actions and environment changes to optimise data capture; whilst learning will need to exploit knowledge gained from observation of past actions and behaviours to predict likely human responses and reactions under different scenarios. During task-performance, assistive systems need to predict the perceptual changes that result as a consequence of human actions. These are challenging tasks which are likely to require step changes in current state of the art capability if they are to be addressed.

The aim of this workshop is to bring together researchers from relevant disciplines to exchange ideas and results on these and related tasks, as well as on the form of existing and future cognitive assistive systems. This will include those working in sensing, such as speech and vision, machine learning and AI, human computer interaction, biomechanics, and on systems and applications, including autonomous platforms, sensor networks and wearable computing, for example. One area in which CA systems are likely to have significant impact is in industrial manufacturing and training, and applications in this area will be of particular interest for this workshop.

Organisers

Dima Damen, University of Bristol, UK

Gabriele Bleser, German Research Center for Artificial Intelligence (DFKI), Germany

Lazaros Nalpantidis, Royal Institute of Technology (KTH), Sweden

Gert Kootstra, Royal Institute of Technology (KTH), Sweden

Renaud Detry, Royal Institute of Technology (KTH), Sweden

Ardhendu Behera, University of Leeds, UK

Luis Almeida, Centre for Computer Graphics, Portugal

Reviewers

Berk Calli, TU Delft, The Netherlands

Sandeep Dubba, University of Leeds, UK

Andrew Gee, University of Bristol, UK

Marc Hanheide, University of Lincoln, UK

Gustaf Hendeby, Linköping University, Sweden

Sinan Kalkan, Middle East Technical University, Turkey

Dirk Kraft, University of Southern Denmark, Denmark

Markus Miezal, German Research Center for Artificial Intelligence (DFKI), Germany

Maja Rudinac, TU Delft, The Netherlands

Norbert Schmitz, German Research Center for Artificial Intelligence (DFKI), Germany

Muralikrishna Sridhar, University of Leeds, UK

Martijn Wisse, TU Delft, The Netherlands

Michael Zillich, TU Wien, Austria

Invited Talks

1 Cognitive Robots with a minimalist action grammar: Theory and Applications

Prof. Yiannis Aloimonos
Computer Vision Laboratory
University of Maryland, United States of America

abstract: There is good reason to believe that humans use some kind of recursive grammatical structure when they recognize and perform complex actions. Following Chomskys Minimalist Program framework, I present a biologically inspired transformational generative grammar of action. In this grammar, action terminals combine hierarchically into temporal sequences of actions of increasing complexity; the actions are bound with the involved tools and affected objects and are governed by certain goals. The tool-role of an object within an action drives the derivation of the action syntax in this grammar and controls recursion, merge and move, the latter being mechanisms that manifest themselves not only in human language, but in human action too. I will demonstrate this grammar in a robotic system that we built to recognize complex human actions involving objects and tools, by integrating Vision and Language. Specifically, the robot automatically constructs a tree structure (and subsequently its semantics) from observations of an actor performing manipulation activities in the real world (such as in the kitchen cutting a tomato, serving coffee, making a sandwich).

*Research supported by the NSF, NIH and the European Union under the Cognitive Systems Program (project POETICON and project POETICON++). Y. Aloimonos is Professor of Computational Vision in the Department of Computer Science, at the University of Maryland, and the Director of the Computer Vision Laboratory in the Institute for Advanced Computer Studies. He studied Mathematics in Athens, Greece and Computer Science in Rochester, NY. He is interested in the integration of vision, action and cognition.

2 Extracting and representing relevant information from high-dimensional data

Prof. Danica Kragic
Center for Autonomous Systems
Royal Institute of Technology (KTH), Sweden

abstract: An assistive system aiming to be truly autonomous must interact with and respond to changes in the environments. To achieve this, a cognitive agent must be capable of extracting relevant information from high-dimensional sensory data. This requires building and maintain models that ensures a reliable interpretations of the surroundings. Centrally to address this challenge is incorporation of a good knowledge representation. It should easily adapt to natural evolvement of the sensory data, be robust to noise while still retaining good representative characteristics as well as being computationally efficient for effective for learning. In this talk, we will present and discuss our view to retain a knowledge representation which satisfies the demands imposed by robotic systems. We will expemplify our approach on several application scenarios.

3 Active sensing and prediction in cognitive robots

Dr. Jeremy Wyatt
Intelligent Robotics Lab
University of Birmingham, United Kingdom

abstract: The ability to predict the effects of actions and to exploit that in planning robust activity is useful in a wide range of cognitive systems. In this talk I'll give a brief overview of work on active sensing and prediction that we are doing in our lab. I'll describe how active sensing can be posed as a problem of planning in information spaces, and in particular on our work on information spaces that evolve probabilistically. This leads us to a variety of algorithms for solving a variety of information state planning problems. These include gaze control for manipulation, object search, and grasping under uncertainty. I'll emphasise how using predictive models is necessary in all these cases, and give some further examples of our work on learning and using prediction. If I have time I'll honestly assess the drawbacks of the presented approaches, and how I hope the field will unfold.

4 Combining Active Vision and Active Touch for Grasping Unknown Objects

Prof. Tamim Asfour
Institute of Anthropmatics and High Performance Humanoid Technologies
Karlsruhe Institute of Technology, Germany

abstract: The ability for grasping objects is a prerequisite for building cognitive situated robots able to act and interact in 24/7 manner in the real world. Such robots should be able to autonomously acquire knowledge about the world and to deal with unknown objects. In this talk, we present our ongoing research towards the implementation of integrated 24/7 humanoid robots able to 1) perform complex grasping and manipulation tasks in a kitchen environment 2) autonomously acquire object knowledge through active visual and haptic exploration and 3) learn actions from human observation and imitate them in goal-directed manner. In particular, the talk will present results on combining visual and haptic information for discovering, segmenting and grasping unknown objects on a humanoid robot. The developed capabilities will be demonstrated on the humanoid robots ARMAR-IIIa and ARMAR-IIIb.

5 Immersive Virtual Manufacturing and Training with Haptic Feedback and Virtual Manikins

Dr. Claude Andriot
Laboratoire de simulation interactive
CEA, France

abstract: This talk is dedicated to immersive virtual manufacturing and training with force feedback and force-controlled virtual manikins. Industrial companies are looking for ways to reduce product design time and costs. The on-going trend is to replace physical prototypes with a "digital mock-up" and to use virtual reality techniques for the validation of the manufacturing, training and maintenance procedures. However, some operations are very difficult to validate without a physical interaction. That is typically the case of assembly /disassembly tasks taking into account not only the manipulated parts but also the human operator.

To address this issue we propose a solution which combines the use of real time advanced mechanical simulation, scale-one interaction with motion tracking and force feedback, and force-controlled virtual manikins.

Thanks to the intuitive use of scale 1 force-feedback, a realistic validation with immediate results can be carried out and human manikins can be animated interactively in the scene, in order to accelerate ergonomic analysis. With the proposed solution the user can control the movements of the manikins precisely, in a much more natural manner than traditional methods based on IK methods.

This new approach is based on several technologies:

- A new real time physical engine XDE dedicated to CAD objects. The XDE software enables users to validate their simulations with real-time (1 Khz) collision detection and advanced mechanical simulation for flexible and rigid objects.
- A large workspace haptic device (6 dof). Most applications demand a large-volume workspace to allow human-scale interaction. Different design approaches aimed at addressing this issue, such as specific non redundant devices, redundant robots, mobile or wearable haptic interfaces and tensed cable architectures. We will present a new device dedicated to this purpose.
- A new framework of online hybrid control for virtual characters, which combines multi-objective control and motion capture techniques. At each time step, the motions of the operator are captured and sent to the character. These captured motions help the character understand what task the operator wants it to perform. Then, the character decides by itself how to execute proper actions while maintaining its balance, in a virtual environment which is different from the real world. Given a manipulation task, the desired virtual task wrench is computed thanks to a force control approach. The control system takes these task wrenches into consideration and solves a constrained quadratic problem for joint torques, which are then used to drive the virtual character. The whole control problem is solved online so that the virtual character can interact in real-time with the virtual environment as well as the operator.

The talk will also present some industrial use cases.

Accepted Papers

Multi-scale cortical keypoints for realtime hand tracking and gesture recognition
M. Farrajota, M. Saleiro, K. Terzic, J.M.F. Rodrigues and J.M.H. du Buf
University of the Algarve, Portugal

Active Perception of Objects for Robot Grasping
J. Bimbo, X. Song, H. Liu, L. Senerivatne, and K. Althoefer
Kings College London, United Kingdom

A Preliminary Account of 3D Visual Search
F. Pirri, M. Pizzoli and A. Sinha
Sapienza University of Rome, Italy

Symbiotic-Autonomous Service Robots for User-Requested Tasks in a Multi-Floor Building
M. Veloso , J. Biswas , B. Coltin , S. Rosenthal, S. Brandao, T. Mericli, and R. Ventura
Carnegie Mellon University, USA
Superior Tecnico, Portugal
Bogazici University, Turkey

Decoupling Behavior, Control, and Perception in Affordance-Based Manipulation
T. Hermans, J. Rehg, and A. Bobick
Georgia Institute of Technology, USA

An in-field magnetometer calibration method for IMUs
S. Salehi, N. Mostofi, and G. Bleser
German Research Center for Artificial Intelligence, Germany

Multi-scale cortical keypoints for realtime hand tracking and gesture recognition

M. Farrajota[1], M. Saleiro[1], K. Terzic[1], J.M.F. Rodrigues[1] and J.M.H. du Buf[1]

***Abstract*— Human-robot interaction is an interdisciplinary research area which aims at integrating human factors, cognitive psychology and robot technology. The ultimate goal is the development of social robots. These robots are expected to work in human environments, and to understand behavior of persons through gestures and body movements. In this paper we present a biological and realtime framework for detecting and tracking hands. This framework is based on keypoints extracted from cortical V1 end-stopped cells. Detected keypoints and the cells' responses are used to classify the junction type. By combining annotated keypoints in a hierarchical, multi-scale tree structure, moving and deformable hands can be segregated, their movements can be obtained, and they can be tracked over time. By using hand templates with keypoints at only two scales, a hand's gestures can be recognized.**

I. INTRODUCTION

Automatic analysis of humans and their actions has received increasingly more attention in the last decade. One of the areas of interest is recognition of human gestures, as these are frequently used as an intuitive and convenient way of communication in our daily life. Recognition of hand gestures can be widely applied in human-computer interfaces and interaction, games, robot control, augmented reality, etc.

In computer vision there are numerous approaches for hand detection, tracking and gesture recognition, although to the best of our knowledge none is really biologically inspired. Kim et al. [7] presented a method for hand tracking and motion detection based on a sequence of stereo color frames. Bandera et al. [1] presented an approach to recognize gestures which are composed of tracked trajectories of different body parts, where each individual trajectory is described by a set of keypoints. Gestures are characterized through global properties of the trajectories which are involved. Suk et al. [17] explored a method for recognizing hand gestures in a continuous video stream based on a dynamic Bayesian network.

Holte et al. [5] presented an approach to invariant gesture recognition using 3D optical flow in a harmonic motion context. Employing a depth map as well as an intensity image of a scene, they used the latter to define a region of interest for the relevant 3D data. Their gesture recognition is based on finding a 3D version of optical flow which results in velocity-annotated point clouds. These are represented efficiently by introducing motion context. The motion context is transformed into a view-invariant representation by applying spherical harmonic basis functions, which yields a harmonic motion context representation. Finally, a probabilistic classifier is applied to identify which gesture best describes a string of primitives. Shen et al. [15] proposed a new visual representation for hand motions based on motion divergence fields, which can be normalized to gray-scale images. Salient regions detected by the MSER algorithm (Maximum Stable Extremal Regions) are then identified in the motion divergence maps. From each detected region, a local descriptor is extracted which captures the local motion pattern.

Our approach is similar to that of [10] in terms of simplicity, with hand tracking although we do not apply color segmentation. A recent development is the "Haar Cascade" for detecting e.g. eyes, mouths, noses and faces [18], [9], also for tracking hands [2]. Algorithms are already included in OpenCV and they are very fast because they employ Haar wavelets, but these wavelets only coarsely resemble Gabor wavelets which are used to model cortical simple cells in area V1.

Recently we presented cortical models based on multi-scale line/edge and keypoint representations, also with keypoint annotation [4], [12], [14]. These representations, all based on responses of simple, complex and end-stopped cells in cortical area V1, can be integrated for different processes: visual reconstruction or brightness perception, focus-of-attention (FoA), object segregation and categorization, and object and face recognition. The integration of FoA, region segregation and object categorization is important for developing fast gist vision, i.e., which types of objects are about where in a scene. We also developed an initial model for cortical optical flow based on keypoints [4]. Experiments have strengthened the idea that neurons in a specialized region of the cerebral cortex play a major role in flow analysis [21], that neuronal responses to flow are shaped by visual strategies for steering in 3D environments [20], and that flow processing has an important role in the detection and estimation of scene-relative object movements [19].

In this paper we present a biologically-inspired method for tracking deformable objects based on keypoints extracted from cortical end-stopped cells. We focus on human hands and gestures which is necessary for joint human-robot manipulation of objects on top of a table: pointing and grasping etc. Our contributions are a realtime cortical hand detector, a new tracking and gesture recognition algorithm, and significantly faster keypoint annotation and tracking algorithms. The advantage of using annotated keypoints is that they provide more information than mere point clouds. The disadvantage is that the filtering involved is very expensive in terms of

[1]Vision Laboratory, LARSyS, University of the Algarve, 8005-139 Faro, Portugal {mafarrajota,masaleiro,kterzic,jrodrig,dubuf}@ualg.pt

CPU time, hence keypoint detection has been implemented on a GPU. The rest of this paper is organized as follows. In Section II we explain keypoint detection and annotation, and in Section III optical flow computation. Hand tracking is explained in Section IV, and we conclude with a discussion in Section V.

II. MULTI-SCALE KEYPOINT ANNOTATION

Keypoints are based on cortical end-stopped cells [12]. They provide important information because they code local image complexity. Moreover, since keypoints are caused by line and edge crossings, detected keypoints can be classified by the underlying vertex structure, such as K, L, T and + shaped junctions, and the angles can be employed. This is very useful for most if not all matching problems: object recognition, stereo disparity and optical flow. In this section we briefly describe the multi-scale keypoint detection and annotation processes.

Recently the original model [12] has been improved such that multi-scale keypoints can be detected in realtime. The improvements concern several important aspects: (1) a new approach to merging keypoints resulting from single- and double-stopped cell responses improves precision at coarse scales; (2) instead of applying many convolutions with filter kernels tuned to many scales and orientations, a Gaussian pyramid is used and all filters are applied in the frequency domain (FFT), which speeds up enormously keypoint extraction at coarse scales; (3) subpixel localization is used, which improves precision at fine scales and partially compensates the loss of precision at coarse scales caused by using the Gaussian pyramid; and (4) a scale selection mechanism is introduced, which significantly reduces the number of duplicated keypoints across scales. These improvements are detailed in a forthcoming paper. Below we briefly describe the algorithms.

The basic principle for multi-scale keypoint detection is based on Gabor quadrature filters which provide a model of cortical simple cells [12]. In the spatial domain (x, y) they consist of a real cosine and an imaginary sine, both with a Gaussian envelope. Responses of even and odd simple cells, which correspond to real and imaginary parts of a Gabor filter, are obtained by convolving the input image with the filter kernel, and are denoted by $R^E_{s,i}(x, y)$ and $R^O_{s,i}(x, y)$, s being the scale, i the orientation ($\theta_i = i\pi/N_\theta$) and N_θ the number of orientations (here 8) with $i = [0, N_\theta - 1]$. Responses of complex cells are modeled by the modulus $C_{s,i}(x, y)$. As mentioned before, there are two types of end-stopped cells, single and double. These are applied to $C_{s,i}$ and are combined with tangential and radial inhibition schemes in order to obtain precise keypoint maps $K_s(x, y)$. For a detailed explanation with illustrations see [12]. Below, the scale of analysis s will be given by λ, the wavelength of the Gabor filters, expressed in pixels, where $\lambda = 1$ corresponds to 1 pixel.

In order to classify any detected keypoint, the responses of simple cells $R^E_{s,i}$ and $R^O_{s,i}$ are analyzed, but now using $N_\phi = 2N_\theta$ orientations, with $\phi_k = k\pi/N_\theta$ and $k = [0, N_\phi - 1]$. This means that for each of the 8 simple-cell orientations on $[0, \pi]$ there are two opposite analysis orientations on $[0, 2\pi]$, e.g., $\theta_1 = \pi/N_\theta$ results in $\phi_1 = \pi/N_\theta$ and $\phi_9 = 9\pi/N_\theta$. This division into response-analysis orientations is acceptable according to [6], because a typical cell has a maximum response at some orientation and its response decreases on both sides, from 10 to 20 degrees, after which it declines steeply to zero; see also [3].

Classifying keypoints is not trivial, because responses of simple and complex cells, which code the underlying lines and edges at vertices, are unreliable due to response interference effects [3]. This implies that responses must be analyzed in a neighborhood around each keypoint, and the size of the neighborhood must be proportional to the scale of the cells. The validation of the line and edge orientations which contribute to the vertex structure is based on an analysis of the responses of complex cells $C_{s,i}(x, y)$. At a distance of λ, and for each direction ϕ_k, responses in that direction and in neighboring orientations ϕ_{k+l}, with $l = \{-2, -1, 0, 1, 2\}$, are summed with different weights equal to $1/2^{|l|}$. After this smoothing and detection of local maxima, each keypoint is then annotated by a descriptor of 16 bits which codes the detected orientations. In the case of keypoints caused by blobs with no underlying line and edge structures, all 16 bits are zero.

This method is an improvement of the previous method [4]. It provides a more detailed descriptor of the underlying line and edge structures, with a significant increase in performance and with a negligible loss of precision. The first five images in Fig. 1 illustrate keypoint detection and annotation at the given scales. For more illustrations see [12].

III. OPTICAL FLOW

Keypoint detection may occur in cortical areas V1 and V2, whereas keypoint annotation requires bigger receptive fields and could occur in V4. Optical flow is then processed in areas V5/MT and MST, which are related to object and ego motion for controlling eye and head movements.

Optical flow is determined by matching annotated keypoints in successive camera frames, but only by matching keypoints which may belong to a same object. To this purpose we use regions defined by saliency maps. Such maps are created by summing detected keypoints over all scales s, such that keypoints which are stable over scale intervals yield high peaks. In order to connect the individual peaks and yield larger regions, relaxation areas proportional to the filter scales are applied [12]. Here we simplify the computation of saliency maps by simply summing the responses of end-stopped cells at all scales, which is much faster and yields similar results. Figure 1 (bottom-right) illustrates a saliency map.

We apply a multi-scale tree structure in which at a very coarse scale a root keypoint defines a single object, and at progressively finer scales more keypoints are found which convey the object's details. Below we use five scales: $\lambda = [4, 12]$ with $\Delta\lambda = 2$. All keypoints at $\lambda = 12$ are supposed to represent individual objects, although we know that it is

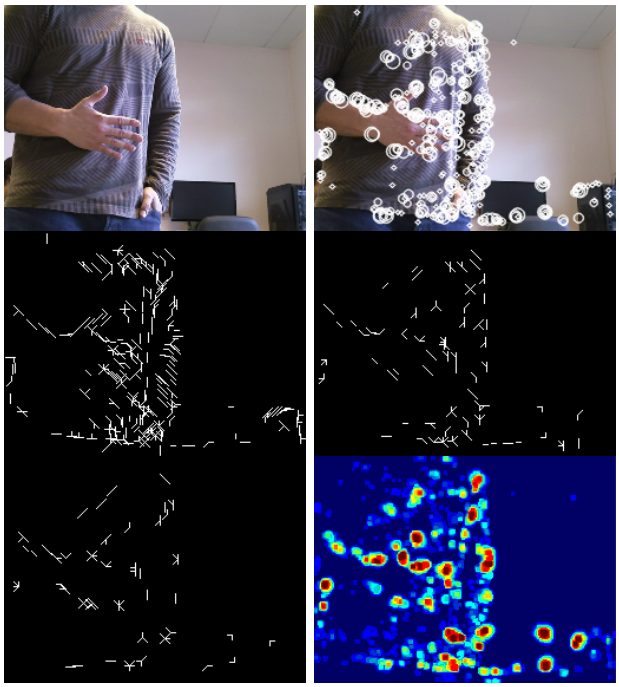


Fig. 1. Left to right and top to bottom: input frame, keypoints detected at all 5 scales, keypoint annotation at scales $\lambda = 4$, 8 and 12, and the frame's saliency map where red indicates higher and blue lower saliency.

possible that several of those keypoints may belong to a same object. Each keypoint at a coarse scale is related to one or more keypoints at one finer scale, which can be slightly displaced. This relation is modeled by down-projection using grouping cells with a circular axonic field, the size of which (λ) defines the region of influence; see [4].

As mentioned above, at a very coarse scale each keypoint – or central keypoint CKP – should correspond to an individual object [12]. However, at the coarsest scale applied, $\lambda = 12$, this may not be the case and an object may cause several keypoints. In order to determine which keypoints could belong to the same object we combine saliency maps with the multi-scale tree structure.

At this point we have, for each frame, the tree structure which links the keypoints over scales, from coarse to fine, with associated regions of influence at the finest scale. We also have the saliency map obtained by summing responses of end-stopped cells over all scales. The latter, after thresholding, yields segregated regions which are intersected with the regions of influence of the tree. Therefore, the intersected regions link keypoints at the finest scale to the segregated regions which are supposed to represent individual objects.

Now, each annotated keypoint of frame i can be compared with all annotated keypoints in frame $i-1$. This is done at all scales, but the comparison is restricted to an area with radius 2λ instead of λ at each scale in order to allow for larger translations and rotations. In addition, (1) at fine scales many keypoints outside the area can be skipped since they are not likely to match over large distances, and (2) at coarse scales there are less keypoints, λ is bigger and therefore larger distances (motions) are represented there. The tree structure is built top-down, but the matching process is bottom-up: it starts at the finest scale because there the accuracy of the keypoint annotation is better. Keypoints are matched by combining three similarity criteria with different weight factors:

(a) The distance D serves to emphasize keypoints which are closer to the center of the matching area. For having $D = 1$ at the center and $D = 0$ at radius 2λ, we use $D = (2\lambda - d)/2\lambda$ with d the Euclidean distance (this can be replaced by dynamic feature routing [4], [13]).

(b) The orientation error O measures the correlation of the attributed orientations, but with an angular relaxation interval of $\pm 2\pi/N_\theta$ applied to all orientations such that also a rotation of the vertex structure is allowed. Similar to D, the summed differences are combined such that $O = 1$ indicates good correspondence and $O = 0$ a lack of correspondence. Obviously, keypoints marked "blob" do not have orientations and are treated separately.

(c) The tree correspondence C measures the number of matched keypoints at finer scales, i.e., at any scale coarser than the finest one. The keypoint candidates to be matched in frame i and in the area with radius 2λ are linked in the tree to localized sets of keypoints at all finer scales. The number of linked keypoints which have been matched is divided by the total number of linked keypoints. This is achieved by sets of grouping cells at all but the finest scale which sum the number of linked keypoints in the tree, both matched and all; for more details see [4].

The three parameters are combined by grouping cells which can establish a link between keypoints in frame $i-1$ and i. Mathematically we use the similarity measure $S = \alpha O + \beta C + \gamma D$, with $\alpha = 0.4$ and $\beta = \gamma = 0.3$. These values were determined empirically. The candidate keypoint with the highest value of S in the area (radius 2λ) is selected and the vector between the keypoint in frame $i-1$ and the matched one in frame i is computed. Remaining candidates in the area can be matched to other keypoints in frame i, provided they are in their local areas. Keypoints which cannot be matched are discarded. Figure 2 (top two rows) illustrates a sequence of 10 frames with a moving hand with detected optical flow vectors.

IV. HAND TRACKING AND MOTION RECOGNITION

Moving objects are segregated and detected by analyzing the optical flow vectors of their multi-scale tree structures. Only trees with keypoints with sufficiently large vectors (displacements of more than 2 pixels between frames) are considered. Deformable objects can be distinguished from rigid ones because some or even all their multi-scale trees possess different motion vectors, i.e., different directions and/or velocities.

Hands performing gestures are a particular class of deformable objects. Hand and gesture recognition is obtained by using a simple and fast algorithm. This algorithm relates

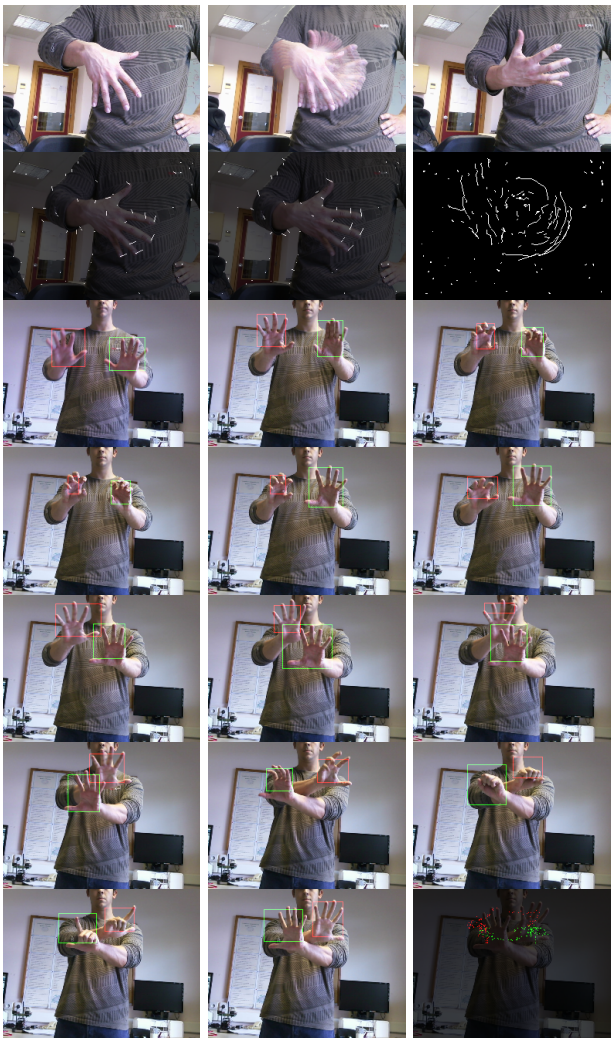


Fig. 2. Top two rows, left to right: initial, combined and final frames of a moving hand sequence; optical flow of two frames; and combined optical flow of the sequence. Bottom five rows: another sequence with tracked hands marked by their bounding boxes. Bottom-right: the combined centers of the boxes.

keypoint positions in previously prepared templates with those detected in acquired image frames. The templates are prepared by simply capuring images of a person with specific hand gestures, after which the hand regions are selected and the keypoint information is stored in small lists; see below. The matching algorithm exploits keypoints at scales not too fine, $\lambda = 8$ and 12, because the number of keypoints is not too big and we are not interested in tiny details. At each scale, and for each template, the angle and the Euclidean distance from each keypoint to all other keypoints are computed. Let us call these primary and secondary keypoints. This yields many but relatively small lists, one for each primary keypoint. Since angles and distances to secondary keypoints are relative to a primary keypoint, all lists are translation and rotation invariant. Typically, a template counts 10 keypoints at scale $\lambda = 8$, such that there are 10 lists each with 9 elements. At scale $\lambda = 12$ there are less. At the moment we only use five templates; see Fig. 3.

Let us first assume that no prior information of a new image frame is available: no known hand position and gesture, and no tracking information. All already available keypoints (because of optical flow) at the two scales and in the entire frame are processed sequentially. In the matching process, the primary keypoint of one template list is positioned at a frame's keypoint, and its secondary keypoints are matched with those in the frame: at positions according to the angles and distances. In order to introduce some flexibility in the matching, for the number of hand-gesture templates cannot be too large, we apply a position tolerance: about 1/5th the size of the template, for example 20×25 pixels in the case of a template of 100×125 pixels. The lists are also mirrored about the major dimension of the template (for the palm and back side of the hand) and rotated by applying only 16 angles because of the position tolerance. Hence, each list involves checking 32 keypoint configurations, or typically $2 \times 10 \times 32$ lists per template, but the matching is fast because a discrete lookup table is used and both the lists and the lookup table are in the CPU's cache memory. When at least 50% of all keypoints in a template list match those in the neighborhood of a frame's keypoint, at one of the two scales, the matching template determines the hand's gesture, its position is known as is its bounding box.

Translation and rotation invariance are obtained by considering (rotated and mirrored) relative angles and distances between keypoints. In order to also achieve scale (size) invariance in the future, each gesture must be represented by several templates captured with different hand sizes (hand-camera distances). A larger number of sizes results in more reliable detection, but costs more CPU time. However, the additional cost is rather low because it only involves matching of many but very small keypoint lists.

By combining optical flow with the hand-gesture detector, hands can be tracked and recognition becomes more robust and faster. Tracking is achieved by combining the last valid hand template, its position in the last frame, and the actual optical flow. This reduces false positives and speeds up the tracking process. At the beginning of the process, camera frames are processed until at least one hand has been detected. Then, the processing of the following frames is simplified by searching the area(s) around the position(s) of the last detected hand(s). Nevertheless, the remaining part of the frames must be analyzed because a hand can be temporarily occluded or a new one can enter the frame. However, this can be done once or twice per second.

For final gesture recognition we assume that the person keeps his hand stable at about the same position, such that the optical flow of the hand is zero or very small. This can be a few camera frames, i.e., a fraction of a second, but it depends on the application: in human-robot interaction while manipulating objects on a table, only occasional final gestures like pointing and grasping are important, but in a game a continuous stream of positions and gestures may be required. The bottom five rows in Fig. 2 show a sequence

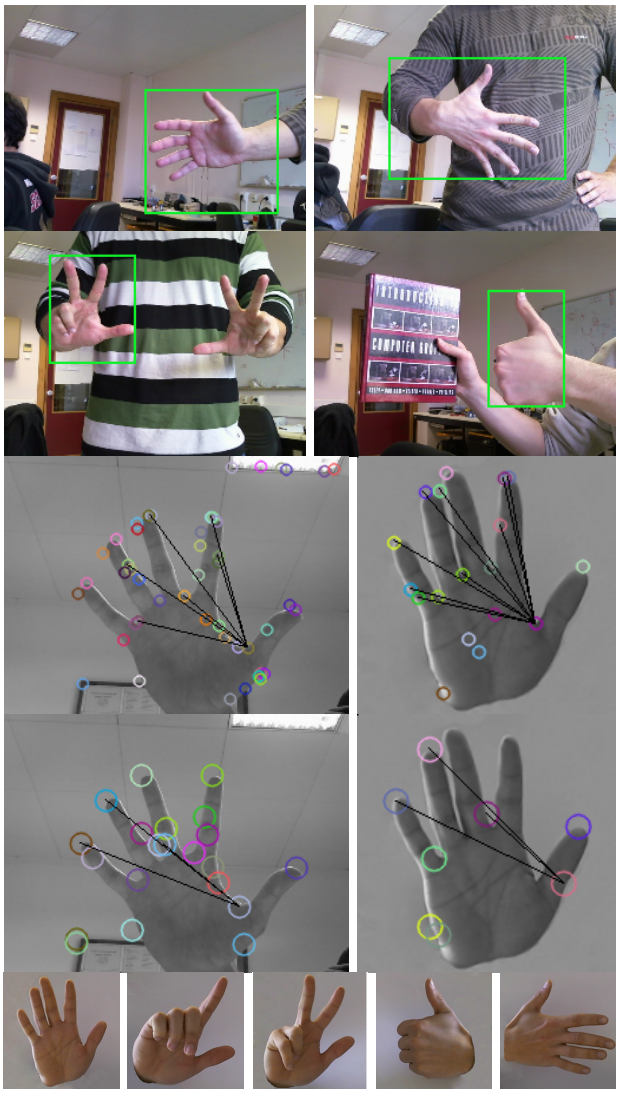


Fig. 3. Top two rows: four examples of recognized gestures. Middle two rows: template matching at $\lambda = 8$ (3rd row) and $\lambda = 12$ (4th row). Bottom: the five hand templates used.

Fig. 4. The optical flow model applied to a person after fetching a bottle from the floor. The sequence shows vectors between successive frames. The two bottom images show the combined vectors while straightening (left), followed by bringing the arm and bottle close to the body and moving the head (right). Significant motions attributed to tracked, segregated regions are indicated by the red arrows.

with two tracked hands. The bottom-right image combines the centers of the bounding boxes. Even if the hands are very close or partly overlapping, the tracking process can separate them. Figure 3 shows recognized gestures (top), the matching process at two scales (middle), and the five templates (bottom). Our method can also be applied to track other deformable objects, for example human bodies; see Fig. 4. This figure shows a sequence of frames while a person straightens after picking up a bottle, and then brings his arm with the bottle close to the body while also straightening his head. In contrast to hand gestures, templates of body gestures remain to be developed and applied.

V. DISCUSSION

In this paper we presented a biologically inspired method for hand detection, with tracking and gesture recognition. After optimizing the keypoint-detection algorithm and by limiting the number of scales, the method works in realtime when using a webcam, and it yields good results despite the fact that color information has not yet been used. The method was expected to work well because of our previous experience with cortical models: the keypoint scale-space provides very useful information for constructing saliency maps for Focus-of-Attention (FoA), and faces can be detected by grouping facial landmarks defined by keypoints at eyes, nose and mouth [12]. In [14] we have shown that the line/edge scale-space provides very useful information for face and object recognition. Obviously, object detection and recognition are

related processes, with a seamless integration in the cortical so-called where and what pathways, i.e., the dorsal pathway (where is it?) and ventral one (what is it?). However, there may be no clear dichotomy in the sense that keypoints are only used in the where pathway and lines and edges only in the what pathway.

Since local clusters of keypoints are mostly related to individual moving objects, object segregation can be achieved and objects can be tracked. Cortical areas MT and MST are involved in optical flow and in egomotion, but recent results obtained with fMRI showed no clear neural activity in their ventral (what) and dorsal (where) sub-areas. Instead, there is elevated activity in between the sub-areas [16]. This might indicate that optical flow at MT level is processed separately or involves both pathways. The fact that the use of only keypoints can lead to very good results in optical flow and object (hand) segregation and tracking may indicate some "preference" of the dorsal (where) pathway for keypoints. This idea is strengthened by the fact that area MT also plays a role in the motion-aftereffect illusion [8], which is tightly related to motion adaptation and prediction.

Being a biologically inspired model, keypoint detection involves filtering input frames with many kernels (complex Gabor functions). We apply eight orientations but only a few scales in order to achieve realtime processing when using a normal webcam: five scales for optical flow and region segregation, of which only two scales are used for hand and gesture detection. The main limitation is the Gabor filtering with keypoint detection. The improved algorithm has already been implemented on a GPU, allowing to process at least 10 frames/s with a maximum resolution of 600×400 pixels and using at least 6 scales if they are not too fine. The GPU's memory of 1 GByte is the bottleneck for using larger images and fine scales because of the Gaussian pyramid.

Ongoing research focuses on motion prediction, a process which occurs in cortical area MST. In addition, instead of only extrapolating hand positions, also the gestures can be tracked and extrapolated, such that the number of templates to be matched can be reduced. Nevertheless, although currently a few distinct gestures are being used, extrapolation may involve more "intermediate" gestures and therefore templates. The ultimate goal is to apply a 3D hand model in the entire process. This can be done by employing cheap and off-the-shelf solutions like a Kinect [11] or two webcams with a biological disparity model. The same applies to human bodies: the tracking and prediction of body joints by exploiting all spatio-temporal information.

ACKNOWLEDGEMENTS

This work was partially supported by the Portuguese Foundation for Science and Technology (FCT) project PEst-OE/EEI/LA0009/2011, EU project NeuralDynamics FP7-ICT-2009-6 PN: 270247, FCT project Blavigator RIPD/ADA/109690/2009 and by FCT PhD grants to MF (SFRH/BD/79812/2011) and MS (SFRH/BD/71831/2010).

REFERENCES

[1] J.P. Bandera, R. Marfil, A. Bandera, J.A. Rodríguez, L. Molina-Tanco, and F. Sandoval. Fast gesture recognition based on a two-level representation. *Pattern Recogn. Lett.*, 30(13):1181–1189, 2009.

[2] A.L.C. Barczak and F. Dadgostar. Real-time hand tracking using a set of cooperative classifiers based on haar-like features. In *Research Letters in the Information and Mathematical Sciences*, pages 29–42, 2005.

[3] J.M.H. du Buf. Responses of simple cells: events, interferences, and ambiguities. *Biol. Cybern.*, 68:321–333, 1993.

[4] M. Farrajota, J.M.F. Rodrigues, and J.M.F. du Buf. Optical flow by multi-scale annotated keypoints: A biological approach. *Proc. Int. Conf. on Bio-inspired Systems and Signal Processing (BIOSIGNALS 2011), Rome, Italy, 26-29 January*, pages 307–315, 2011.

[5] M.B. Holte, T.B. Moeslund, and P. Fihl. View-invariant gesture recognition using 3D optical flow and harmonic motion context. *Comput. Vis. Image Underst.*, 114(12):1353–1361, 2010.

[6] D.H. Hubel. *Eye, Brain and Vision*. Scientific American Library, 1995.

[7] H. Kim, G. Kurillo, and R. Bajcsy. Hand tracking and motion detection from the sequence of stereo color image frames. *Proc. IEEE Int. Conf. on Industrial Technology*, pages 1–6, 2008.

[8] A. Kohn and J.A. Movshon. Neuronal adaptation to visual motion in area MT of the macaque. *Neuron*, 39:681–691, 2003.

[9] R. Lienhart and J. Maydt. An extended set of haar-like features for rapid object detection. *Proc. IEEE ICIP*, pages 900–903, 2002.

[10] C. Manresa, J. Varona, R. Mas, and F. Perales. Hand tracking and gesture recognition for human-computer interaction. *Electronic Letters on Computer Vision and Image Analysis*, 5(3):96–104, 2005.

[11] I. Oikonomidis, N. Kyriazis, and A. Argyros. Efficient model-based 3D tracking of hand articulations using kinect. *Proc. BMVC*, pages 101.1–101.11, 2011.

[12] J. Rodrigues and J.M.H. du Buf. Multi-scale keypoints in V1 and beyond: object segregation, scale selection, saliency maps and face detection. *BioSystems*, 2:75–90, 2006.

[13] J. Rodrigues and J.M.H. du Buf. A cortical framework for invariant object categorization and recognition. *Cognitive Processing*, 10(3):243–261, 2009.

[14] J. Rodrigues and J.M.H. du Buf. Multi-scale lines and edges in V1 and beyond: brightness, object categorization and recognition, and consciousness. *BioSystems*, 95:206–226, 2009.

[15] X. Shen, G. Hua, L. Williams, and Y. Wu. Dynamic hand gesture recognition: An exemplar-based approach from motion divergence fields. *Image and Vision Computing (In Press)*, 2012.

[16] A. Smith, M. Wall, A. Williams, and K. Singh. Sensitivity to optic flow in human cortical areas MT and MST. *European J. of Neuroscience*, 23(2):561–569, 2006.

[17] H. Suk, B. Sin, and S. Lee. Hand gesture recognition based on dynamic Bayesian network framework. *Pattern Recogn.*, 43(9):3059–3072, 2010.

[18] P. Viola and M.J. Jones. Rapid object detection using a boosted cascade of simple features. *Proc. IEEE CVPR*, 1:511–518, 2001.

[19] P.A. Warren and S.K. Rushton. Optic flow processing for the assessment of object movement during ego movement. *Current Biology*, 19(19):1555–1560, 2009.

[20] K.P. William and J.D. Charles. Cortical neuronal responses to optic flow are shaped by visual strategies for steering. *Cerebral Cortex*, 18(4):727–739, 2008.

[21] R.H. Wurtz. Optic flow: A brain region devoted to optic flow analysis? *Current Biology*, 8(16):R554–R556, 1998.

Active Perception of Objects for Robot Grasping

Joao Bimbo, Xiaojing Song, Hongbin Liu, Lakmal D. Senerivatne, and Kaspar Althoefer

***Abstract*— This paper presents an overview of a system which, combining information from a number of different sensors and applying a number of perception strategies, is able to recognise an object's surface material, predict at what threshold the object will slip, prevent this slippage and track a grasped object's pose. For the classification and slip prediction, the robot needs to stroke the object twice and find out the static and dynamic parameters of the interaction between the instrumented finger tip and the object surface. A method is also presented which, taking information from a vision system and the contact locations, is able to estimate the pose of the object with respect to the hand. An integrated approach of these methods is proposed and discussed.**

I. INTRODUCTION

Robot grasping and manipulation require accurate and real-time information of the physical interaction between the object and the fingers. Among the most important information that needs to be acquired are the friction properties of this interaction and the knowledge of the object's location within a robotic hand.

Although for most robot grasping applications the Coulomb friction model is used, which simplifies the slippage phenomenon to the calculation of the static and dynamic friction coefficients, it is known that the break-away force ratio – the ratio between tangential and normal force at slip occurence – is dependent on the changes in velocity and acceleration between the object and the fingers [1]. To better understand the occurence of slip, one has to use more sophisticated friction models that take these dynamics into account. This paper presents the application of the LuGre Model [2] to robot grasping in order to predict slip more adequately and compensate it before the onset of slip [3]. The knowledge of these parameters is shown to also enable a system to classify the surface material, as it is a very specific property of each material [4].

To track an object's position during a manipulation task, the usage of a vision system alone may not provide accurate and timely information about an object's pose, due to occlusions, the object being outside the camera's range or the computational costs of a robust real-time vision tracking. Hence, a system's vision sensing can be complemented with more information that not only allows faster updates but also improves the object's tracking beyond the accuracy of the camera. This paper presents a method that combines vision with tactile sensing which refines the knowledge of the object pose from grasping by fitting its model to the current contact locations, providing an estimate of the object's current pose with very low latency.

Integrating these different strategies together enables to close the loop between perception and action, by preventing slip and compensating it while still keep track of the object's current location. This paper presents this integration strategy first by describing the hardware and software architecture and the sensing algorithms in Sections II and III, then detailing the friction model and its application in surface classification and slip prediction in Section IV. The pose estimation algorithm is presented in Section V and the integration method is proposed in Section VI. Section VII presents the conclusions of this integrated system.

J. Bimbo, X. Song, H. Liu, L.D. Senerivatne and K. Althoefer are with the Centre for Robotics Research, King's College London, WC2R 2LS London, United Kingdom, {joao.bimbo, song.xiaojing, hongbin.liu,lakmal.senerivatne k.althoefer}@kcl.ac.uk

II. SYSTEM ARCHITECTURE

The hardware used on the grasping system consisted of a Mitsubishi™ RV6-SL 6 Degree of Freedom robotic arm with a BH8 BarrettHand™ attached on the end-effector with ATI Nano17 6-axis force-torque sensors on each of the three fingers, and a Microsoft Kinect™ camera located opposite and facing the robot workspace. Two computers were connected through an Ethernet network with one of the computers connected to the camera through the USB port and the other computer connected to the ATI sensors through National Instruments™ PCI Data Acquisition cards and to the robotic hand through a serial port.

The software was implemented in C++ using the ROS [5], [6] platform. The results were analysed in Matlab™.

III. ARTIFICIAL FINGER TIP

A. Design

A customised finger tip was designed and attached to each finger, consisting of an ATI Nano17 6-axis force-torque sensor, a plastic cylinder attached to the sensing side of the sensor and a hemispherical plastic tip on the other end. The dimensions were accurately known and the choice of an hemispherical shape was made so that the surface could be easily parametrisable and differentiable. A photo of the resulting system is shown in Fig. 1.

B. Contact Location

The contact location on the finger tip is determined using the algorithm proposed in *Bicchi et al* [7] where, by using the force and torque measurements, the contact centroid on a known and parametrisable convex surface can be determined. This contact sensing based on the net force acting on a body is typically known as "intrinsic contact sensing". The method for finding this unique solution has been implemented and

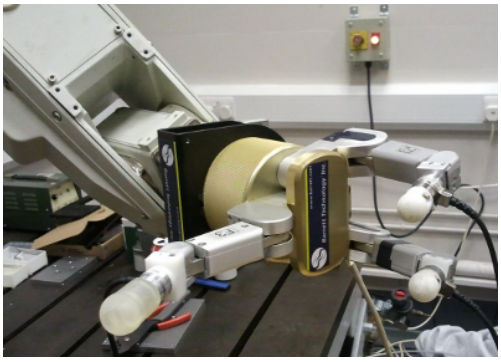


Fig. 1. Barrett hand with customised finger tips

validated by *Liu et al* [8] who, besides finding the contact location used this information to determine the normal and tangential components of the acting net force along with the "normal torque" – the torque acting along the contact normal direction. This algorithm proves to be very accurate when dealing with a single contactlocation or a continuous surface, rending a root mean square error of 266 μm and its frequency was set to 100 Hz. Fig. 2 shows a screenshot of an application developed to visualise this data in real time.

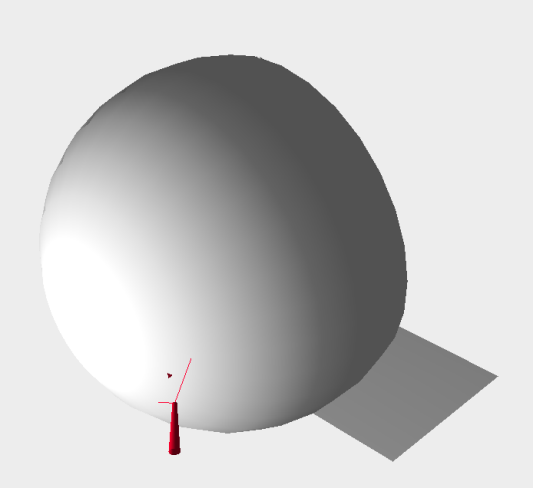


Fig. 2. Intrinsic sensing visualiser – the red cone represents the net force acting on the hemispherical finger tip, the lines represent the normal and tangential components and the arrow shows the local torque

IV. SURFACE PROPERTY IDENTIFICATION

A. Friction Model

Most robot grasping applications address the determination of the friction effect using the Coulomb friction model, which is a simplification of the friction phenomenon that requires only the calculation of two constants, the static and kinetic coefficients, μ_s and μ_c, which are dependent only on the applied load (normal force) and the surface materials. However, later research has shown that the ratio of the friction and normal forces at slip occurrence, known as break-away friction-ratio (BF-ratio) is not a constant, varying with changes in velocity and acceleration [1].

The chosen model to describe the friction effect was the LuGre model, which models the asperity of two contacting surfaces as elastic bristles. Equations (1) to (3) detail the LuGre model, where z is the bristles' average deflection, σ_0, σ_1 and σ_2 are constant stiffness, damping and viscosity coefficients. F_n and F_t are the normal and tangential forces, v is the sliding velocity and v_0 is the Stribeck coefficient [2], [3].

$$\dot{z} = v - \sigma_0 \frac{|v|}{s(v)} z \tag{1}$$

$$s(v) = F_n(\mu_c + (\mu_s - \mu_c)e^{-|\frac{v}{v_0}|^2}) \tag{2}$$

$$F_t = \sigma_0 z + \sigma_1 \dot{z} + \sigma_2 F_n v \tag{3}$$

B. Surface Classification

The knowledge of the friction parameters described in the previous Section is very useful for the choice of applied forces by the fingers during grasping and manipulation tasks, but their determination can also be useful to identify the surface material of an object, as they are very specific properties of each material.

A method to recognise the surface material using the friction parameters was presented by *Liu et al* [4] which uses haptic exploration to find out the friction parameters and a supervised learning algorithms to distinguish the material. This strategy consists of gently sliding a surface with the instrumented finger tip, first increasing and then decreasing the sliding speed with low accelerations, obtaining the relation between the friction force and the velocity [4].

Given that the variations in velocity are low, the LuGre model presented in Section IV-A can be simplified to its *quasi-static* form, described in (4), where the symbols have the same meaning as in (1), (2) and (3).

$$F_t = \text{sgn}(v)[\mu_c + (\mu_s - \mu_c)e^{-|\frac{v}{v_0}|^2}] + \sigma_2 F_n v \tag{4}$$

After the estimation of these parameters, a training set with the properties of different types of materials was created and tested against other samples. Several supervised learning algorithms were tested to classify these new samples, where the naive Bayes classifier proved to be the most accurate as well as more computational efficient. This combination of the *quasi-static* LuGre friction model and a naive Bayes classifier was shown to provide sufficient parameters to accurately discriminate different materials, reaching an overall accuracy of 88.5%, even between surfaces with very similar friction properties such as aluminium and brass or glass and ceramic with a minimum success rate of 79.2%.

C. Slip Prediction

Most robot manipulation tasks require accurate detection of slippage phenomena and mechanisms that can predict and prevent this occurrence. In order to successfully predict the onset of slip the dynamics of the contact should be modelled. This modelling makes use of the LuGre model described in Section IV-A, where both static and dynamic parameters

need to be estimated. The estimates of static parameters are taken from the surface classification step described in Section IV-B, and only the dynamic parameters need to be estimated.

In order to obtain the dynamic parameters σ_0 and σ_1 a second stroke is performed, this time with higher acceleration (10 mm/s^2), with increasing and then decreasing speeds. The nonlinear Levenberg-Marquardt (LM) method was used to identify the parameters. The hysteresis loop for increasing and decreasing velocities is plotted in Fig. 3, where the finger accelerates to 13 mm/s and decelerates back to stationary, at a rate of 10 mm/s^2.

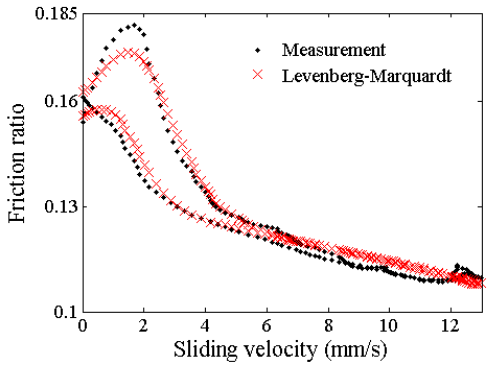


Fig. 3. Friction ratio (F_n/F_t) estimation using Levenberg-Marquardt and Extended Kalman Filter against real measurements [3]

V. OBJECT POSE ESTIMATION

A. Vision

The vision system consisted of a Microsoft Kinect camera observing the robot workspace. This camera provides depth as well as colour information. Then, a method to detect the object, create its 3D model and save it to a database was employed [9]. The model was then used for the vision to track the object [10]. This method relies on visible features on the object and may fail if the object is occluded or if not enough features are visible by the camera.

B. Pose Rectification

Due to the mentioned limitations and the computational demand to visually track the object in real time, a method was created to use the tactile data acquired by the finger tips to correct the pose of the object and have a good estimate of the object's position in real time. This method, presented by Bimbo *et al* [11], relies on the object model point cloud and the contact locations to find a transform that makes the object coincide with the contact locations on the finger tips, increasing its robustness with the number of independent contacts. The algorithm tries to find a set of parameters – a translation vector and a rotation quaternion – that transforms the contact locations to match the object surface, minimising the distance from finger to object. Once found, the inverse transform is applied to the object point cloud.

This strategy allows for the robot system to track the object in real time even when the vision system is not able to and also refine the estimation given by the machine vision method. Fig. 4 shows a picture of an initial estimate of the object and the resulting fitted position of the object.

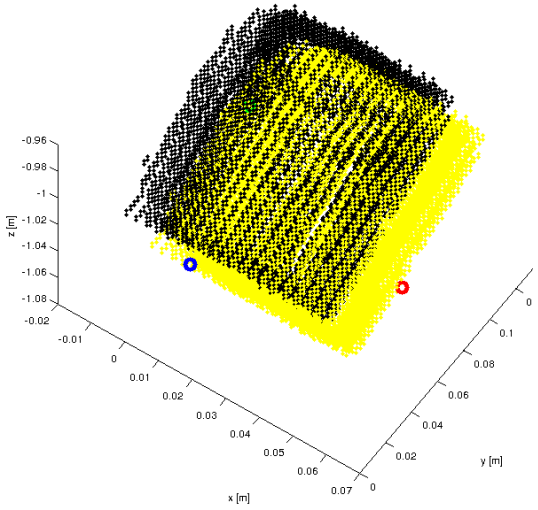


Fig. 4. Pose correction algorithm simulation result – original object point cloud in black, corrected position in yellow and contact locations in red, blue and green circles

VI. ACTION-PERCEPTION

A. Slip Compensation

With the knowledge of the friction parameters of the finger-object interaction, a real-time slip predictor was implemented that took into account not only the acting forces but also velocity and acceleration changes and disturbing forces on the object to calculate the break-away force-ratio to define a threshold for the onset of slip and a safety margin of 5% to trigger a compensator, which would command the hand controller to increase the grasping force, thus preventing slippage. Fig. 5 plots the behaviour of the system, where it can be seen that when the current friction ratio enters the safety margin it compensates by increasing the grasping force, reducing the ratio F_n/F_t to safe values. The response is usually fast enough that it prevents the slippage.

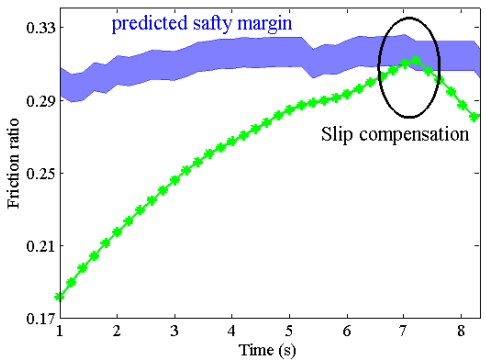


Fig. 5. Slip prediction and compensation (Figure taken from [3])

B. Object Tracking

The combination of the different sensing hardware and techniques presented on this paper can provide a closed-loop system where, not only the contact location on the finger is known, but also this location with respect to the object can be estimated and suitable grasping forces can be devised according to the current friction threshold.

The proposed system architecture, outlined in Fig. 6, begins by acquiring the object model and estimating its pose, then, if enough fingers are contacting the object, rectifies the first pose estimate acquired by the camera as explained in Section V-A. If the grasp is stable and there is no slippage or movement of the fingers, the object location will follow the arm kinematics , as there is no relative movementbetween the object and the hand. In case the objects come close to slippage, the fingers will have to grip harder on the object, possibly changing its position and orientation, therefore requiring the pose correction algorithm described in Section V-B. If slippage is detected, *i.e.* if the friction ratio goes above the estimated threshold before the compensation mechanism is able to react, the same behaviour follows, correcting the pose estimation through the use of contact locations. At any moment, if the vision is able to track the object, this loop will restart with the new estimate given by the vision system.

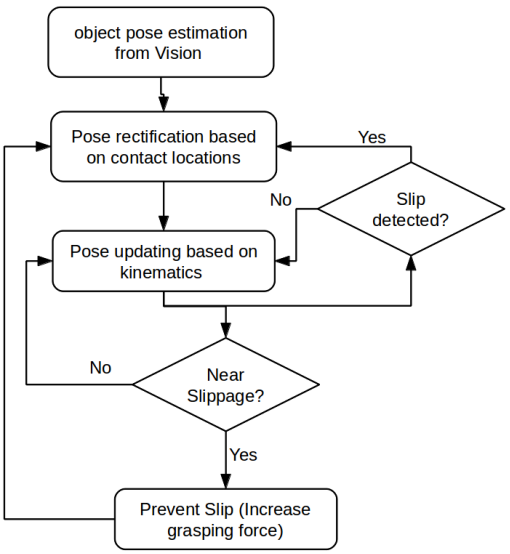


Fig. 6. Active perception and reaction flowchart

VII. CONCLUSIONS

The methods presented in this paper aim to improve the perception of a robotic system by enhancing the knowledge of a grasped object's surface properties, enabling the system to accurately predict the onset of slip and the object's current location within the hand. An active exploration method is presented that, by stroking a surface twice can retrieve the surface friction parameters, classify the surface material and estimate the break-away friction ratio. Another method is used to estimate the object's current location inside the hand by taking into account a vision obtained model and the position of the contact locations. These methods are integrated into a system that, during grasping and manipulation tasks can avoid slippage and, in case of an unexpected slip or a change in the grasping pose can still track the estimated position of the object.

ACKNOWLEDGMENT

The research leading to these results has been funded by the HANDLE European project (FP7/2007-2013) under grant agreement ICT 231640 – http://www.handle-project.eu

REFERENCES

[1] H. Olsson, K. Astrom, C. de Wit, M. Gafvert, and P. Lischinsky, "Friction models and friction compensation," *Eur. Journal of Control*, vol. 4, no. 3, pp. 176 – 195, 1998.

[2] C. C. de Wit and H. Olsson, "A new model for control of systems with friction," *IEEE Transactions on Automatic Control*, vol. 40, no. 3, pp. 419–425, 1995.

[3] X. Song, H. Liu, J. Bimbo, K. Althoefer, and L. D. Seneviratne, "A Novel Dynamic Slip Prediction and Compensation Approach Based on Haptic Surface Exploration," in *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'2012)*, to be published.

[4] H. Liu, X. Song, J. Bimbo, L. Seneviratne, and K. Althoefer, "Object Surface Material Recognition through Haptic Exploration using a Intelligent Contact Sensing Finger," in *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'2012)*, to be published.

[5] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Ng, "ROS: an open-source Robot Operating System," 2009.

[6] Willow Garage, "Robot Operating System." [Online]. Available: www.ros.org

[7] A. Bicchi, J. K. Salisbury, and D. L. Brock, "Contact Sensing from Force Measurements, Tech. Rep. 3, June 1993.

[8] H. Liu, X. Song, J. Bimbo, and K. Althoefer, "Intelligent Fingertip Sensing for Contact Information Identification," in *Proc. of the Second ASME/IEEE International Conference on Reconfigurable Mechanisms and Robots (ReMAR 2012)*, 2012.

[9] N. Burrus, M. Abderrahim, J. Garcia, and L. Moreno, "Object Reconstruction and Recognition leveraging an RGB-D camera," in *Proc. of MVA2011 IAPR Conference on Machine Vision Applications*, 2011.

[10] Deliverable D23 of HANDLE European Project, "Visual and Tactile Perception System: Evaluation Report," Tech. Rep.

[11] J. Bimbo, S. Rodriguez-Jimenez, H. Liu, X. Song, N. Burrus, L. D. Seneviratne, M. Abderrahim, and K. Althoefer, "Object Pose Estimation and Tracking by Fusing Visual and Tactile Information," in *Proc. of 2012 IEEE International Conference on Multisensor Fusion and Information Integration (MFI 2012)*, to be published.

A Preliminary Account of 3D Visual Search

Fiora Pirri, Matia Pizzoli and Arnab Sinha*

***Abstract*— Visual search is a very difficult task when pursued in complex real world environments, and it is still not clear how humans can accomplish it with relative ease. The human visual system can achieve it while other tasks are performed in parallel, such as talking and walking and, in particular eye movements combine with stimuli and neural activity in parallel in this pre-attentive task. Among other things, this shows that the search activity does not absorb the whole neural processing, as humans do not fully analyze the whole scene, and merge top-down information with bottom-up stimuli to make up a strategy guiding the search. This ability is at the basis of human ecological behaviors, and awareness of other beings while reciprocating with them. Despite a great amount of contributions on visual attention in autonomous systems, research on pre-attentive modeling, involved in visual search, is still in its infancy. In this paper we provide a preliminary model of visual search based on an experimental set up that allows us to collect the information gathered by a subject performing visual search tasks of different targets in a natural environment. The information available in the visual field of the subject and, in particular, what the subject has been observing, together with her pose and the object pose with respect to the subject, is used in the attempt to analyze and interpret the correlation between the target and the spots that drive the subject towards the target. Despite we do not consider top-down cues, we show some interesting results on predicting salient spots when either the whole information – as gathered by the experiments – is available and when only partial information is available.**

I. Introduction

Visual search in real world environments is a hot topic in computer vision and robotics, as it involves attention, ability to scan the environment and a search strategy selecting only those spots in the visual field that can either lead to the target or be the target. The research in visual attention has taken advantage of eye-tracking experiments [1]–[8] and has proposed different computational models, the best known being the ones introduced in [9], [10], while a recent review can be found in [11]. Computational models of visual attention have been applied in several cognitive robotic applications (see [12] for a survey), such as tracking [13], simultaneous localization and mapping [14] and exploration [15]. Nonetheless, very little has been done to model the process of visual search in natural, 3D environments based on observations of the human searching strategy. Indeed, the problem of visual search is relevant in its own and crucial for all robot vision applications, as it amounts to picking up only those regions in space that might contain the target. The issue is rather involving since it requires to model, at the same time, 3D saliency and the relation between what is salient and the target, and it requires to face the motion of the subject in a 3D environment.

In this paper we introduce a methodology for estimating a visual search strategy based on an experimental setting. More precisely, we collect data from several experiments in which humans perform different search tasks. This collection amounts to a data structure mentioning the video, the depth map relative to the subject, the localization of the subject in a global map and the regions around the point of regard (POR) projection to the 3D scene and its appearance. Here the POR is *the point on the retina at which the rays coming from an object regarded directly are focused*.

The first step to estimate a search strategy is the acquisition of the observations: the subject performs a search task while wearing a device that projects the point of regard in space, tracking the scan path in the reconstructed environment [8]. This phase allows to pinpoint the foveated, high resolution regions of the spotted objects.

The second step consists in collecting the points of regard into bundles of 3D point clouds. These bundles, when transformed into a collection of features, are space-time tensors that, in principle, collect all the interesting regions of the explored space by different subjects undergoing the same experiment.

The collected data offer the ground truth of a process which ends successfully, namely when all the targets have been found. We show that if these data are divided into a training and a testing set, an optimal classifier is able to match all the attended PORs with less than 10% of error, under the proviso that classification is done with one of the chosen targets, in the same environment and on the same acquisition video. These conditions are too restrictive; furthermore, when the visual search task is performed by a robot, only part of the data collected in the experiment is available. We thus put, in this paper, the bases to define a prediction model for visual search, by identifying how to complete the missing data. This is just a preliminary step towards a thorough understanding of how to replicate visual search in autonomous systems.

The paper is organized as follows. Section II introduces the measurement framework. In Section III we identify a set of properties in the experiment useful to derive a search strategy, which is then proposed in Section IV. Finally, in Section V, some preliminary results are presented and in Section VI the conclusion is discussed.

II. POR localization and mapping and the search measurements method

In this section we introduce the methods used to determine the value of all the measurable properties of the visual search

*The authors are with the ALCOR lab, Department of Computer and System Engineering, Sapienza University of Rome, via Ariosto 25, 00185, Rome, Italy. {pirri,pizzoli,sinha} at dis.uniroma1.it

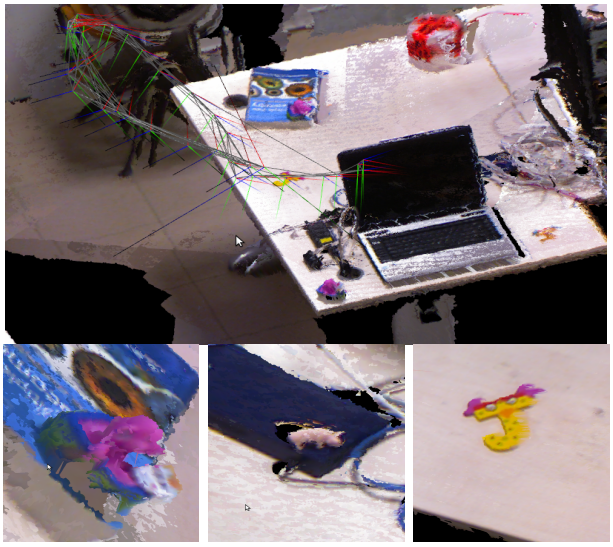


Fig. 1: Collection of 3D data for the search experiment. Top: recovery of the subject's pose and 3D structure. Bottom: objects used in the search experiment.

task via the determination of the point of regard (POR), namely the projection in space of the point on the retina where the image of the observed object is formed.

An experiment consists of searching a number of targets of the same type[1]. The subject can observe a sample of the targets which are then disposed in the room in different places, possibly partially hidden.

To record and take measures of the visual search process several collateral quantities need to be considered: the eye movements via the pupil center, the head movements, the pose of the subject in space, the three-dimensional structure of the attended scene where the POR region can be individuated.

Actually other measures are involved, that here we shall implicitly assume, concerning the relevance of the attended spot in space according to eye velocity and incidence of neighbor spots.

A. Subject localization

The Gaze Machine framework (GM) has been introduced to build a map of attended three-dimensional regions within an experiment [8]. The GM acquisition device is worn by the subject and allows the recording of the scene (the stimuli) by a calibrated stereo rig and the tracking of the subject's gaze by means of a pair of fast cameras directed to the subject's eyes. The subsequent analysis of the collected data comprises: 1) estimating the 3D POR π^c in the reference frame of the GM scene cameras; 2) estimating the 3D pose (6 degrees of freedom) of the GM scene cameras in a *world* reference frame for the experiment, in terms of translation $\mathbf{t}$ and orientation $\mathbf{R}$; 3) projecting the gaze direction (the counter direction of the rays towards the retina) in the *world* reference frame as $\pi^w = \mathbf{R}\pi^c + \mathbf{t}$.

[1]Contrasting or mimetic objects, with respect to the scene, have been used, as illustrated in Figure 1.

The GM calibrated stereo rig records the stimuli, allowing for dense 3D reconstruction from multiple views. The scene is acquired at a rate of 30 fps and the association with the much faster acquisition of gaze, over 120 fps, is maintained by time-stamping. From the acquired scene, stereo visual localization ([16], [17]) is computed to estimate the pose of the subject in the 3D experimental setting. Most of the issues affecting the localization of a camera system (see, for example, [18]) also apply to the GM, with some notable differences due to the importance played by head movements: saccades larger than 30 degrees are precursive of head movements. Therefore a hasty eye-head movement, although problematic for localization, needs to be tracked for an accurate description of the visual search properties.

To face this problem we have labeled coherent sequences, with respect to the visual field, by key frames. More precisely, let $\mathscr{M}^G$ be the current map constructed so far and let $(\mathbf{x}_i, \mathbf{X}_i)$, $i = 1, \ldots N$, the N pairs of matched image and map points. The linear algorithm [19] for exterior orientation is used as the core of a RANSAC-based, robust estimation of the pose of the camera projecting the 3D points $\mathbf{X}_i$ as $\mathbf{x}_i$. The error function takes into account re-projection errors in both the image planes of the stereo pair. Using the l and r superscripts to identify quantities for left and right cameras, and assuming the relative pose $\mathbf{R}^s$ and $\mathbf{t}^s$ of the cameras known from calibration, the error function is:

$$\begin{aligned} \varepsilon_i \quad &= d\left(\mathbf{K}^l\mathbf{R}(\mathbf{X}_i+\mathbf{t}), \mathbf{x}_i^l\right)^2 + \\ &\quad d\left(\mathbf{K}^r\mathbf{R}^s[\mathbf{R}(\mathbf{X}_i+\mathbf{t}) - \mathbf{t}^s], \mathbf{x}_i^r\right)^2. \end{aligned} \tag{1}$$

Here d is the Euclidean distance and $\mathbf{K}$ is the 3×3 matrix of intrinsic parameters. A final Levemberg-Marquardt optimization refines the linearly estimated pose by minimizing (1). Upon the acquisition of a new pair of scene frames, the pose of the subject is estimated from matched features among the current frames and the current map $\mathscr{M}^G$. This method guarantees a global consistency across the whole experiment and it is accurate as long as the global map is accurate.

B. Keyframe selection

When no knowledge of the pose and point of regard of the subject is retained because of some large saccade, it is necessary to suspend the map refinement build so far, initiate a new map $\mathscr{M}'$ and eventually connect it with the previous one when the visual fields get superimposed, which necessarily happens due to the continuity of the head-body motion of the subject.

Let $\mathscr{K}_t$ be the last determined keyframe, at time t. A motion characterized by a small baseline between the current frame $Im_{t+\Delta t}$ and $\mathscr{K}_t$ is best described by a 2D homography $\mathbf{H}$. On the other hand, the fundamental matrix $\mathbf{F}$ provides the geometry of general camera motions [18]. Building on the Geometric Robust Information Criterion (GRIC, [20]), a score function Γ is evaluated for both the $\mathbf{F}$ and $\mathbf{H}$ motion models to obtain their fit with the data. The score function takes into account the n matched features with the last keyframe $\mathscr{K}_t$, the residuals e_i, the number k of model

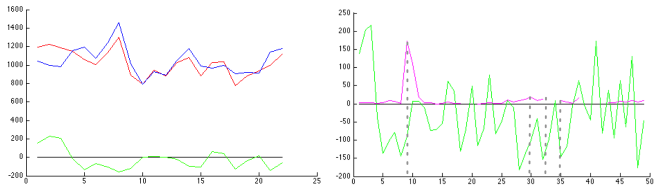


Fig. 2: Keyframe selection criterion. Left: $\Gamma(\mathbf{F})$ (red), $\Gamma(\mathbf{H})$ (blue) and $\Gamma(\mathbf{F})-\Gamma(\mathbf{H})$ (green). Right: $\Gamma(\mathbf{F})-\Gamma(\mathbf{H})$ (green) and δ (magenta). Keyframes are selected in correspondence of dashed lines.

parameters, the error standard deviation σ, the dimensions r of the data and the dimension q of the model, as follows:

$$\Gamma = \sum_{i=1}^{n} \rho(e_i^2) + [nq\ln(r) + k\ln(rn)], \tag{2}$$

where $\rho(e_i^2) = \min\left(\frac{e_i^2}{\sigma^2}, 2(r-q)\right)$.

Equation (2) returns the lowest score for the model that best fits the data. The selected motion model is used to evaluate the gaze variation. Let γ and γ' be the gaze projection in the left and right frames, then a new keyframe is instantiated on the basis of the following criterion (see Figure 2):

$$(\Gamma(\mathbf{F})-\Gamma(\mathbf{H}))\cdot\delta<0, \quad \delta=\begin{cases} \gamma'^{\top}\mathbf{F}\gamma & \text{if } \Gamma(\mathbf{F})<\Gamma(\mathbf{H}) \\ ||\mathbf{H}\gamma-\gamma'|| & \text{otherwise.} \end{cases} \tag{3}$$

C. POR mapping

Upon the instantiation of a new keyframe at time $t+\Delta t$, the following steps are performed.

Subsequence Optimization Sparse Bundle Adjustment (SBA) [21] on the sequence of the last k camera poses and the set of unoptimized matches using the re-projection error introduced in eq. (1) as objective function, with the j, here, indexing $\mathbf{X}_j$:

$$\begin{aligned} &\min_{\mathbf{R}_i,\mathbf{t}_i,\mathbf{X}_j}\textstyle\sum_{ij}\varepsilon_{ij} \\ &\varepsilon_{ij} = d\left(\mathbf{K}^l\mathbf{R}_i(\mathbf{X}_j+\mathbf{t}_i),\mathbf{x}_{ij}^l\right)^2 + d\left(\mathbf{K}^r\mathbf{R}^s[\mathbf{R}_i(\mathbf{X}_j+\mathbf{t}_i)-\mathbf{t}^s],\mathbf{x}_{ij}^r\right)^2. \end{aligned} \tag{4}$$

Here $i=t-1,\ldots,t-k$, $\mathbf{x}_{ij}^c$, $c\in\{l,r\}$ is the point $\mathbf{X}_j$ imaged by the i-th left or right camera respectively.

Map Upgrade The global structure $\mathscr{M}^G$ is updated with the set of the newly optimized points $\mathscr{P}$: $\mathscr{M}^G=\mathscr{M}^G\cup\mathscr{P}$.

Subsequence Initialization $k=0$, $\mathscr{P}=\emptyset$.

When a new keyframe is selected, the previous subsequence can be either suspended or terminated. In the latter case the correspondent points and cameras are optimized and the resultant structure is added to the global map.

III. Measurable properties of visual search

In this section we focus on the feature set, obtained by an experiment E, needed to verify whether the collected data structure is a basic support for studying visual search. In other words, we want to verify if all the data collected in a visual search experiment E offer a good description of the problem, and we do so using the values of all the measurable properties of the visual search for learning. Indeed, these data are a good approximation of all the measurable visual properties of a visual search task[2] if a good classifier is able to learn a function f such that $y=f(\mathbf{x})$ will issue a 1 if $\mathbf{x}$ is the vector of data of an attended POR, and -1 otherwise. We shall see in Section IV that, when visual search involves artificial attention with autonomous systems, we cannot access all the measurable properties of visual search, as these data are available only in an experiment E performed by a human.

Let E_T be the data collected in a visual search experiment, $T>0$ the time lapse of the experiment; let J be a set of indices, agreeing with the time stamp of drawn measures. Then the structure obtained by dense matching and triangulation of the frames, together with the frames themselves, is defined to be

$$\mathscr{S}_T = \langle\{\mathscr{I}_j(x,y)\}_{j\in J},\{\mathscr{M}_j^G(X,Y,Z)\}_{j\in J},\{\mathscr{M}_j^O(x^\star,y^\star,z^\star)\}_{j\in J}\rangle. \tag{5}$$

Here, for each image $\mathscr{I}_j(x,y)$ in the sequence, denoting the intensity value at pixel location $(x,y)^\top$, its corresponding 3D global map is $\mathscr{M}_j^G(X,Y,Z)$ and relative, w.r.t. the observer, map is $\mathscr{M}_j^O(x^\star,y^\star,z^\star)$, where non measured points are specifically indicated.

From the set $\mathscr{S}_T$ we infer the feature space in input to classification. An observation is thus defined by the q-dimensional vector $\mathbf{X}_{jk}$, $j\in J, k\le mn$, with mn the size of $\mathscr{I}$, collecting the following information. Let $\mathscr{I}_j(x,y)$ be an image in the sequence and $\mathscr{M}_j^G(x,y,z)$ and $\mathscr{M}_j^O(x^\star,y^\star,z^\star)$ the global and relative maps, then the following features specify $\mathbf{X}_{jk}$:

- the time stamp, namely the time in *ms* at which the image $\mathscr{I}_j$ has been taken;
- the intensity, at pixel location $\mathbf{x}=(x,y)^\top$, as the sum and the product of its 8-neighbors;
- the position $(x^\star,y^\star,z^\star)^\top$ in $\mathscr{M}_j^O$ of the attended pixel $(x,y)^\top$, relative to the observer, and its distance $d_\mathbf{O}$ from the observer;
- the position $(X,Y,Z)^\top$ in $\mathscr{M}_j^G$ of the attended pixel $(x,y)^\top$, in global coordinates, with origin the starting position of the visual search task;
- the value at $(x,y)^\top$ of the second order image derivatives $(\frac{\partial^2}{\partial x^2}\mathscr{I}(x,y), \frac{\partial^2}{\partial xy}\mathscr{I}(x,y), \frac{\partial^2}{\partial y^2}\mathscr{I}(x,y))$;
- the distances $D=(d_1,\ldots,d_5)^\top$ between $(x,y)^\top$, and the position of up to 5 PORs $\mathbf{p_1},\ldots,\mathbf{p_5}$ on the image plane. This step is related to the construction of proto-objects that we do not mention here.

The collected set of observations is denoted $\mathscr{W}$, of size $mnT\times q$. A training set and a test set are formed by sampling from $\mathscr{W}$, the positive instances being the attended regions in the scene while the negative instances are sampled from the unattended regions. The described set of features has been selected from a wider set, essentially derived from $\mathscr{S}$, using a specific feature selection algorithm based on an ℓ_1-norm

[2]The notion of visual properties does not mention all the measurable properties related to the neural processing measurable from brain activity.

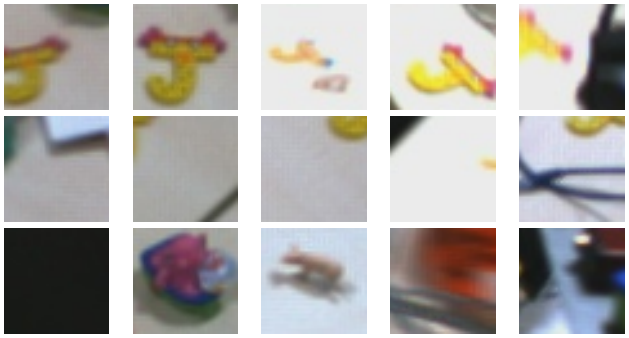


Fig. 3: image patches for the "FOUND J" class (first row), the "Looking for J" class (second row) and the "Distractors" class (third row).

classifier that we do not report here. For a search experiment in which the subjects were asked to search yellow "J" letters in the lab, we classify the PORs in three classes, namely, "FOUND J" (FJ), "Looking for J" (LJ), and "Distractors" (D). This labeling is done manually and with respect to the following *subjective* or *perceptual* understanding of the individual labeler.

Found J (FJ): when a POR is close to the target.

Looking for J (LJ): when the viewer is looking on a plane, e.g. a desktop, the floor, a book, where it is highly probable to find the target.

Distractors (D): when the viewer *is not* looking for J, and possibly distracted by some highly salient object in the scene. When the viewer is looking at the walls or distracted by other objects then also the corresponding PORs are classified as distractors: they correspond to tasks other than the searching for J. In Figure 3 the image patches corresponding to few samples from these three classes are shown.

In table I, we show the correlation of image patches from classes FJ and D, with that of class LJ. The correlations

TABLE I: Correlation between several patches from class LJ (Looking for J) with patches from classes FJ (Found J) and D (Distractor)

LJ	1st	2nd	3rd	4th	5th
FJ	−0.27	−0.21	0.36	−0.18	0.13
D	−0.05	0.06	0.06	−0.12	−0.05

shown in table I, and the images in figure 3 make clearer that the distractors are not correlated with the image patches corresponding to looking for J (LJ), whereas the target (class FJ) has more correlation with the looking for J image patches.

IV. Spot prediction: toward an autonomous search strategy

So far we have presented the methods that led us to suitably classify the data, collected for visual search of targets, distributed in a real world environment. These methods provide a completely new insight in the problem of visual search and they can be summarized as follows:

1) What is attended while performing visual search in a real world environment.
2) How to classify what is attended in terms of distractors, random spots, and regions correlated with the search, as explained in Section III.
3) How to relate the visual search with the subject localization, namely where, which and when some regions are repeatedly looked at and how they are meaningful to single out from these data a search strategy.

Given these properties with their input features space, as specified in Section III, an optimal classifier, such as the SVM, is able to classify a video of a human visual search. Results are given in Section V. The results are meaningful if the visual search is achieved in the same environment and the target is one of those searched by the human.

The goal, when the visual search is performed by an autonomous system, is to minimize the space to be analyzed and measured. In this case the available data is a subset of the complete data we have in the GM experiment: an autonomous system can acquire the images, estimate a sparse 3D map and a sparse-based localization. The goal is to predict these data, given what can be learned from the experiments. We can define the prediction problem for visual search as follows.

Visual search prediction problem Given N experiments, performed by a human, and a complete set of measurable properties, suitably measured by a device, not necessarily the GM; given the feature space defined on the these properties, a current set of images and their depth map, according to the robot localization, the prediction problem is to determine which regions in space are either the target or regions correlated to the search.

For example regions correlated to the search could be the floor, or the tables, and the prediction problem tells that the search can predict the regions without requiring to recognize tables or floor. The situation is as follows. Let $\mathbf{X}$ be a random variable following a known multivariate parametric distribution $F_{\mathscr{X}}$. Let $\mathbf{Z}$ be an observed value of $\mathbf{X}$, a sample of the parametric distribution $F_{\mathscr{X}}$, missing some values, that is, $\mathbf{Z} = (z_1, z_2, \circ, \circ, z_5, \circ, \ldots, z_n)^\top$, where the circles stand for the values that cannot be read. $\mathbf{Z}$ is a noisy observation with values strongly affected by noise. Our goal is to infer the missed values, namely, it is to estimate the probability that the missing values $\mathbf{Z}$ should be attended or not, as it is the problem is close to inpainting and restoration [22].

More precisely. Let $f_{\mathscr{X}}$ be the multivariate density, with parameters Θ, modeling the features, specifying the properties of the visual search task. We can assume that we have learned this model from all the complete features sets returned by the experiments. It follows that, similarly to the SVM, that we have seen in Section III, given a complete observation $\hat{\mathbf{X}}$, the density will issue a high probability in correspondence of a region that would be attended in the visual search and that is correlated to the target. When, instead of $\hat{X}$ we have a noisy vector $\mathbf{Z}$ with missing values, the problem is to find those values completing $\mathbf{Z}$ such that the probability is maximal if $\mathbf{Z}$ belongs to a region that would

be attended.

We can note that the missing data are only related to 3D points in global coordinates, and possibly their distance with respect to the observer, and the observation $\mathbf{Z}$ is the vector of features about a pixel (x,y). Let $\Omega = (\Omega_1 \times \ldots \times \Omega_m)$ be the complete domain of these data, the goal is to estimate $p(\mathbf{X}|\mathbf{Z},\Omega)$ based on the missing values observation $\mathbf{Z}$ and the known model of the whole data set, namely $f_{\mathscr{X}}(\mathbf{X}|\Omega)$. Note also that we assume that if the data were complete then the density would almost correctly classify the input, namely we assume that we have an optimal model $f_{\mathscr{X}}$ for the complete data set. The Bayesian problem is thus:

$$p(\mathbf{X}|\mathbf{Z},\Omega) = \frac{p(\mathbf{Z}|\mathbf{X},\Omega) f_{\mathscr{X}}(\mathbf{X}|\Omega)}{p(\mathbf{Z}|\Omega)}. \quad (6)$$

Here, however, we can drop the Ω in $f_{\mathscr{X}}$ and we can define a normalization constant for the denominator in eq. (6) above. To obtain $p(\mathbf{X}|\mathbf{Z},\Omega)$ we need to estimate $p(\mathbf{Z}|\mathbf{X},\Omega)$, since once $\mathbf{X}$ is inferred its value at the prior will be given. We do a two step process. In the first step we draw by Monte Carlo sampling, from the known function $f_{\mathscr{X}}$, with parameters Θ[3], a set of samples among which we retain a set of candidates minimizing the square of the norm, namely:

$$C_{X_Z} = \{\mathbf{Y} \in \Omega |\ \|\mathbf{Z} - \mathbf{Y}_{\setminus b}\|^2 \leq \tau\}. \quad (7)$$

Here the subscript $\setminus b$ indicates that those values in $\mathbf{Y}$ that are in the same position of the missed values in $\mathbf{Z}$, have been blinded to compute the square norm; for example $\mathbf{Z} = (1, \circ, 3, \circ)^\top$ and $\mathbf{Y} = (1.5, 2, 2.7, 4)^\top$ then $\mathbf{Y}_{\setminus b} = (1.5, \circ, 2.7, \circ)^\top$. Now each element in C_{X_Z} is a potential candidate. If $\mathbf{X}$ is the ideal value to be estimated we should have that $\mathbf{Y}_i = \Psi_i \mathbf{X} + \eta_i$. Here η_i is the null space of Ψ_i. However in the presence of several minimizer for

$$(\Psi^\top \Psi)^{-1} \Psi^\top \mathbf{Y} \quad (8)$$

the goal is to choose the one that satisfy the constraints in terms of distance and depth map at the point (x,y) at which the observation $\mathbf{Z}$ is taken. We define a range of possible values of Ω which depend on the neighbor observations. The second step consists in the completion. Let Ω_0 be the range of the missed values of the observation $\mathbf{Z}$ at the pixel (x,y), coherent with its neighbors. Let $\lambda(z)$ be the specified range for the $\mathbf{Y}$, let $\hat{\mathbf{Y}}$ be the mean of the sampled $\mathbf{Y}_i$ and α be the precision matrix of the the $\mathbf{Y}$. Then we have

$$E(\hat{\mathbf{Y}}|\mathbf{X}_z,\Omega) = \alpha \int_{\Omega_0} [(\mathbf{Y} - \Psi\mathbf{X})^2 \lambda(z) + \mu_D + \mu_G] dz. \quad (9)$$

Here μ_D and μ_G are regularization parameters, that must manage the distribution between the distances from the observer and the depth map. From this term and equation (6) is possible to recover the probability that the observation is an attended region or not. Some experiments are illustrated in Section V. We have briefly illustrated preliminary aspects of the model because of space and because we are at an early stage of a full model for predicting visual search.

[3]Note that we do not discuss $f_{\mathscr{X}}$ here, but we can easily assume that it is a mixture of Gaussians, with parameters Θ.

V. Experiments and results

In order to investigate the accuracy of the proposed method in localizing and mapping the PORs, we produced the ground truth data by placing 5 visual landmarks in the experimental scenario and measuring their position with respect to a fixed reference frame. Then, the GM framework has been used to acquire and analyze 6 test sequences in which subjects have been asked to fixate the visual landmarks while freely moving in the scenario, annotating (by speaking) the starting and ending of the landmark observations. In each sequence, about 60 PORs were produced for each landmark. The validation sequences comprise about 6000 frames each. After registration of the subject initial pose with the fixed reference system, the PORs in the annotated frames were computed and compared with the ground truth, producing a Root Mean Square (RMS) value of 0.094 meters.

TABLE II: Evaluation of the accuracy of the proposed POR localization method against a ground truth made of known marker positions. The overall RMS is 0.094.

Landmark	Frames	Fixations	Error mean	Error var.
1	6210	60	0.0112	0.0033
2	6531	65	0.0458	0.0238
3	5997	58	0.2397	0.0215
4	6370	63	0.0491	0.0153
5	6189	60	0.0742	0.0060

A quantitative analysis of the keyframe selection strategy used manual coding to produce the ground truth. After the experiment, the subject was shown the scene sequence overlapped with the POR projection on the image and individuated the coherent subsequences by annotating the keyframes. The performance measure is the *agreement*, defined as the ratio between the number of subsequences detected by the system and those labeled by the subject. The average agreement on sequences yielding 120-200 keyframes was 85%.

TABLE III: Results from the *k-fold cross validation* of the maximum margin classification using the complete image and bundle feature set.

iter.	Pos.	wp_+	wp_-	Acc.
1	44707	0.0127	0.0318	95.334%
2	46881	0.01883	0.0206	93.591%
3	420034	0.0093	0.0157	93.019%

TABLE IV: Experiments with LIBSVM on data completed by the estimation and optimization method.

Exp.	Data	Pos.	Accuracy	False Pos.	False Neg.	BER
1	48661	468	0.87	61	260	0.068
2	185631	2374	0.79	498	1833	0.110
3	53924	521	0.85	77	416	0.079

Given an observation task, a model trained on the complete set of features described in Section III is able to predict if

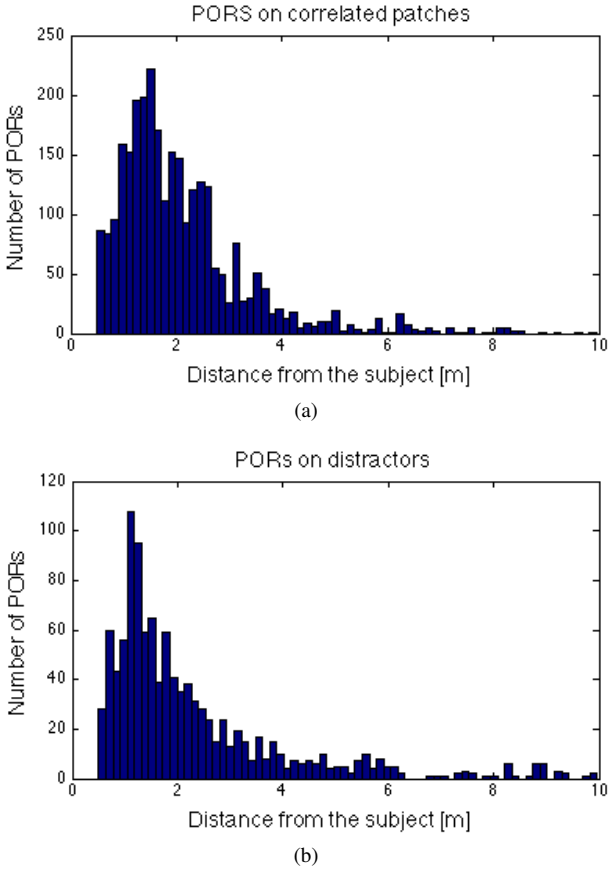


Fig. 4: Distribution of PORs with respect to the distance from the subject for a search task. PORs are labelled, respectively, as *Correlated* (a) and *Distractors* (b).

a new sample point is likely to be attended. To validate this assumption, we ran maximum margin classification experiments. A *K-fold cross-validation* strategy has been followed: we divided the available data comprising more than 6 million points in 3 subsets; in turn, 2 of the three subsets have been used to train the classifier and the remaining one for validation. The process is iterated until every subset is used for validation. As expected, classification accuracy is very high, as reported in Table III.

The influence of a knowledge *a priori* in the "Search J" experiment is confirmed by comparing the distributions of PORs distances in case of correlated patches and distractors (Figure 4). The search is mainly focused on a range of distances compatible with the desktops, while the distractors are spread across the scenario.

Finally, to validate the prediction method proposed in Section IV, maximum margin classification experiments are used on the inferred data set with a Gaussian Kernel. Results are reported in Table IV.

VI. Conclusions

The presented work addresses the problem of deriving a visual search strategy based on data collected in 3D gaze tracking experiments. Training data take into account the visual stimuli, the relative pose of the observer and the attended object in the 3D world, features of both the image and the structure of the attended object. The resulting system represents a completely novel approach to modeling visual search and a first step towards the definition of computational models for attention in real-world scenarios.

Acknowledgment

The research presented in this paper has been funded by the EU-FP7 *NIFTi* project.

References

[1] L. Yu and M. Eizenman, "A new methodology for determining point-of-gaze in head-mounted eye tracking systems," *IEEE Trans. on Biom. Eng.*, vol. 51, no. 10, pp. 1765 –1773, 2004.

[2] S.-W. Shih and J. Liu, "A novel approach to 3-d gaze tracking using stereo cameras," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 34, no. 1, pp. 234 –245, 2004.

[3] Z. Ramdane-Cherif and A. Nait-Ali, "An adaptive algorithm for eye-gaze-tracking-device calibration," *Instrumentation and Measurement, IEEE Transactions on*, vol. 57, no. 4, pp. 716 –723, April 2008.

[4] U. Rajashekar, I. van der Linde, A. Bovik, and L. Cormack, "Gaffe: A gaze-attentive fixation finding engine," *Image Processing, IEEE Transactions on*, vol. 17, no. 4, pp. 564 –573, 2008.

[5] A. Villanueva and R. Cabeza, "A novel gaze estimation system with one calibration point," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Trans. on*, vol. 38, no. 4, pp. 1123 –1138, 2008.

[6] D. Hansen and Q. Ji, "In the eye of the beholder: A survey of models for eyes and gaze," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, no. 3, pp. 478 –500, 2010.

[7] G. Iannizzotto and F. La Rosa, "Competitive combination of multiple eye detection and tracking techniques," *Industrial Electronics, IEEE Transactions on*, vol. 58, no. 8, pp. 3151 –3159, 2011.

[8] F. Pirri, M. Pizzoli, D. Rigato, and R. Shabani, "3d saliency maps," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*, 2011, pp. 9 –14.

[9] J. Tsotsos, S. Culhane, W. Wai, Y. Lai, N. Davis, and F. Nuflo, "Modeling visual attention via selective tuning," *Artifical Intelligence*, vol. 78, pp. 507 – 547, 1995.

[10] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," *IEEE PAMI*, vol. 20, no. 11, pp. 1254–1259, 1998.

[11] S. Frintrop, E. Rome, and H. I. Christensen, "Computational visual attention systems and their cognitive foundations: A survey," *ACM Trans. on Applied Perception*, vol. 7, no. 1, pp. 6:1–6:39, 2010.

[12] M. Begum and F. Karray, "Visual attention for robotic cognition: A survey," *Autonomous Mental Development, IEEE Transactions on*, vol. 3, no. 1, pp. 92 –105, 2011.

[13] V. Mahadevan and N. Vasconcelos, "Spatiotemporal saliency in dynamic scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, pp. 171–177, 2010.

[14] C. Siagian and L. Itti, "Biologically inspired mobile robot vision localization," *Robotics, IEEE Transactions on*, vol. 25, no. 4, pp. 861 –873, 2009.

[15] A. Carbone, A. Finzi, A. Orlandini, and F. Pirri, "Model-based control architecture for attentive robots in rescue scenarios," *Autonomous Robots*, vol. 24, pp. 87–120, 2008.

[16] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry for ground vehicle applications," *Journal of Field Robotics*, vol. 23, p. 2006, 2006.

[17] H. Strasdat, A. Davison, J. Montiel, and K. Konolige, "Double window optimisation for constant time visual slam," in *IEEE International Conference on Computer Vision*, 2011, pp. 2352 –2359.

[18] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. CUP, 2000.

[19] P. Fiore, "Efficient linear solution of exterior orientation," *IEEE PAMI*, vol. 23, no. 2, pp. 140–148, 2002.

[20] P. H. S. Torr, "Geometric motion segmentation and model selection," *Phil. Trans. R. Soc. Lond. A*, vol. 356, no. 1740, pp. 1321–1340, 1998.

[21] M. Lourakis and A. Argyros, "Sba: A software package for generic sparse bundle adjustment," *ACM Transactions on Math. Software (TOMS)*, vol. 36, no. 1, p. 2, 2009.

[22] M. Bertalmío, G. Sapiro, V. Caselles, and C. Ballester, "Image inpainting," in *SIGGRAPH*, 2000, pp. 417–424.

Symbiotic-Autonomous Service Robots for User-Requested Tasks in a Multi-Floor Building

Manuela Veloso[1], Joydeep Biswas[2], Brian Coltin[2], Stephanie Rosenthal[1], Susana Brandao[3,4], Tekin Mericli[5], and Rodrigo Ventura[4]

***Abstract*— Although since the days of the Shakey robot, there have been a rich variety of mobile robots, we realize that there were still no general autonomous, unsupervised mobile robots servicing users in our buildings. In this paper, we contribute the algorithms and results of our successful deployment of a service mobile robot agent, CoBot, in our multi-floor office environment. CoBot accepts requests from users, autonomously navigates between floors of the building, and asks for help when needed in a symbiotic relationship with the humans in its environment. We present the details of such challenging deployment, in particular the effective real-time depth-camera based localization and navigation algorithms, the symbiotic human-robot interaction approach, and the multi-task dynamic planning and scheduling algorithm. We conclude with a comprehensive analysis of the extensive results of the last two weeks of daily CoBot runs for a total of more than 8.7 km, performing a large varied set of user requests.**

I. Introduction

We have been pursuing the goal of deploying multiple autonomous mobile robots capable of performing tasks as requested by users in our multi-floor building. There are several sub-problems to address:

1) Localizing and navigating autonomously and safely
2) Providing an intuitive interface for users to schedule tasks for the robot
3) Scheduling conflict-free task plans for each robot
4) Interacting with humans
5) Overcoming robot limitations to perform tasks

There has been considerable work in the robotics community to solve each of these sub-problems individually as well as combinations of these problems. However, rarely are all of these goals addressed simultaneously on a single platform. The robots Shakey [1], Xavier [2], and museum tour guide robots [3], [4], [5] have addressed some of these problems to varying degrees of success. Additionally, we have been inspired by the contributions of RoboCup@Home [6], a competition for autonomous indoor service robots with a wide scope of human-interaction challenges.

[1]M. Veloso and S. Rosenthal are with the Computer Science Department, Carnegie Mellon University, USA mmv, srosenth at cs.cmu.edu

[2]J. Biswas and B. Coltin are with The Robotics Institute, Carnegie Mellon University, USA bcoltin, joydeepb at cs.cmu.edu

[3]S. Brandao is with the ECE Department, Carnegie Mellon University, USA sbrandao at ece.cmu.edu

[4]S. Brandao and R. Ventura are with the ECE Department, Instituto Superior Tecnico, Lisbon, Portugal rodrigo.ventura at isr.ist.utl.pt

[5]T. Mericli is with Department of Computer Engineering, Bogazici University, Istanbul, Turkey tekin.mericli at boun.edu.tr

Fig. 1. The deployed CoBot-2 service robot.

In this paper, we contribute a complete system that addresses all of these sub-goals to perform tasks requested by the occupants of an office building. In addition to being effective in work environments, such a system has ready applications to assistive care in hospitals or nursing homes, where it could help overburdened nurses and caregivers to deliver items to bedridden patients.

We have developed two robots, CoBot-1 and CoBot-2. The robots, agile in their navigation due to their omnidirectional bases, purposefully include a modest variety of sensing and computing devices, including a controllable camera, a Microsoft Kinect depth-camera, a small Hokuyo LIDAR, and a touch-screen tablet. The CoBots autonomously localize and navigate in the building using depth-camera and LIDAR based localization and navigation algorithms.

Recently, we have effectively deployed CoBot-2 (Figure 1) to the occupants of our building for several hours each day. Occupants can schedule the CoBot robots to perform four different tasks: (i) go to a room, (ii) deliver a spoken message to a room, (iii) transport an object from one room to another, and (iv) escort a person from an elevator to a room. It has been rewarding to witness the robot continuously moving in our building among four floors without supervision. In this paper, we contribute the underlying technical algorithms, as well as the results of the robot's deployment.

The deployed CoBot-2 follows a *symbiotic* autonomy approach [7] given that it has limitations in its perception, cognition, and action. The robot proactively assesses that it needs help and asks humans to help resolve its limitations, particularly the physical ones. For example, the CoBot robots

do not have arms, and therefore they ask for help for manipulating objects and pressing elevator buttons.

CoBot-2 is deployed to perform tasks requested by occupants of the building known as "task solicitors", and gets help from other humans around the robot known as "task helpers". Figure 2 illustrates the multiple components of our symbiotic CoBot robot. Task solicitors can request tasks on an online web server which are then processed by a *scheduler*, which determines their execution time. The *scheduler* sends these tasks to the *task planner and executor* that divides tasks into autonomous actions and symbiotic interactions to request help to ride the elevator and to manipulate objects.

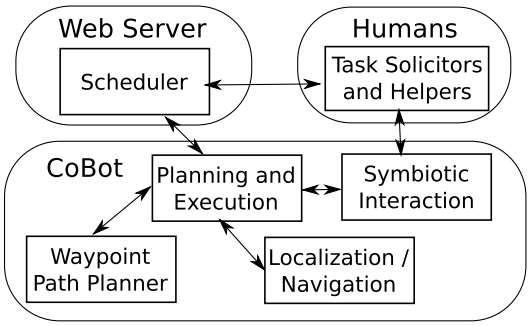


Fig. 2. Connections between web server, CoBot and humans.

In this paper, we present our approach to long-term task-centered robot deployment and extensive results from our own CoBot robots. In particular, we contribute:

1) A web based scheduler that accepts users requests and generates a conflict-free schedule for the robot (Section 2)
2) A task execution framework that allows the robot to overcome its limitations by asking for help from humans (Section 3)
3) A complete autonomous robot system capable of performing its scheduled tasks (Section 4)
4) A set of results from deployment of the robot to the occupants of the building (Section 5)

II. Scheduling User-Requested Tasks

Users request tasks over the web for CoBot-2 to perform. From CoBot-2's website, registered users can book a new task, view a list of their own scheduled tasks and completed tasks, cancel scheduled tasks, and see the current position of the robot.

When scheduling a task, users choose:

- *Task Type*. The task request. The options include 1) go to a room, 2) deliver a spoken message, 3) transport an object from one location to another, and 4) escort a visitor from the elevator to an office.
- *Task Parameters*. Task-specific options. This includes the destination location(s) for each task. Other options could include a spoken message to deliver or the name of an object to transport.
- *Time Constraints*. When the task should be executed. The user can specify "as soon as possible", a specific time, or a window of time.

Each request submitted by the user is sent to the *scheduler*, which may either accept or reject it. If the scheduler rejects a request, it proposes alternative times when the robot is available. The scheduler must fit the requested tasks into a *schedule*, a mapping of tasks to execution times. To plan the schedule, we represent each task request T_i as a tuple $T_i =< s, e, l^s, l^e, d >$. Task T_i must begin execution within the time interval $[s, e]$ where s and e are times in seconds (all time constraints are converted internally to windows of time). The task T_i is expected to take d seconds to complete from start to finish. The task duration d does not include the time taken to travel from the ending location of one task to the starting location of the next, this is estimated by a function $c(l_1, l_2)$ which estimates the time taken to travel between two locations. The task begins at the location $l^s \in L$ and ends at the location $l^e \in L$, where L is the set of all named locations CoBot-2 can travel to, including offices, kitchens and lounges.

A. Estimating Task Times

To effectively produce a schedule of tasks for the robot to execute, we must estimate both how long each task will take and how long it will take CoBot-2 to travel from one task to the next. This is the role of the Task Time Estimator. There are three components of the task duration d to consider:

- t_{nav} - *Navigation Time.* The time spent moving from one location to another. This is computed from the distance planned from CoBot-2's navigation graph and its velocity. It does not include use of the elevators.
- $t_{elevator}$ - *Elevator Transportation Time.* The time spent waiting for and using the elevator.
- t_{task} - *Task Specific Time.* The time specific to the task. For example, when delivering a spoken message, t_{task} includes the time taken to recite the message.

The expected task execution time is the sum

$$d = t_{nav} + t_{elevator} + t_{task}\,.$$

Similarly, the expected time, or cost, to travel between two locations includes the same components, except for t_{task}.

$$c(l_1, l_2) = t_{nav} + t_{elevator}$$

This cost is an estimate, and the actual time to execute a task may be either longer or shorter. CoBot-2 will adjust its schedule accordingly during execution.

B. Forming a Schedule

Based on the estimated execution and travel times, the scheduler fits all of the tasks into a schedule, a mapping of tasks to planned execution times. This is a variant of the Dynamic Vehicle Routing Problem with time windows, in which a fleet of vehicles must visit a set of locations, each within a certain time interval [8]. In the batch problem, the scheduler is given a list T of n task requests that the robot must fulfill by finding a set of starting times t_i so that no two tasks overlap and each task is fulfilled within the requested time window $[s_i, e_i]$. Each task has an estimated duration d_i which does not include travel time between tasks, since

this depends on the tasks' ordering. The function c gives the estimated cost to travel between two locations, as computed by the waypoint path planner.

We solve for the variables t_i with a mixed integer program (MIP). Our first set of constraints states that each time t_i must fall within the start and end times of the window: $\forall i \ s_i \leq t_i \leq e_i$. Next, we add constraints to ensure that no two tasks overlap, handling both the case that task i comes before task j and the reverse order. We introduce helper indicator variables $pre_{i,j}$ which indicate whether task i occurs before task j.

$$\forall i, j \ t_i + d_i + c(l_i^e, l_j^s) - t_j \leq |e_i - s_j|(1 - pre_{i,j})$$
$$\forall i, j \ t_j + d_j + c(l_j^e, l_i^s) - t_i \leq |e_j - s_i| pre_{i,j}$$

We minimize the sum of the starting times $\sum_i t_i$ to ensure that user tasks are completed as soon as possible.

Solving an MIP is NP-hard; however, for smaller problem instances it can be done relatively quickly. We generated a thousand problem instances of fifteen tasks with two minutes to a half hour duration, with randomly generated time windows over the course of a four hour period. We expect CoBot to receive similar patterns of requests in the real world. The scheduler solved over 99% of the problems in under two seconds on a laptop computer. In the few cases where a solution is not found in two seconds, we can reject the user's request and ask them to modify their time window.

In practice, the task requests are not processed in a batch, but come in an online-fashion over the web. We reschedule everything whenever a new request is made. If a schedule cannot be found, the user's request is rejected and the user has an opportunity to relax their constraints.

After a schedule is formed, it is sent to the robot's task planner to execute. The task planner provides continuous feedback regarding the state of execution, and informs the scheduler when the task has been completed.

III. Planning and Executing Tasks

CoBot-2 is capable of autonomous localization and navigation, but cannot manipulate objects and has a limited ability to perceive the elevators. However, users can request tasks such as transporting objects, that require these capabilities. Task executor divides each task into autonomous actions and symbiotic interactions to seek help with actions it cannot perform autonomously.

The task executor first plans symbiotic interactions with its task solicitors at the start and end of the task to place and remove objects and confirm task completion. The task solicitors are expected to be willing to help because they requested the task to be performed and therefore want it to succeed. Then, the executor uses waypoint path planning to determine the lowest cost path to travel to rooms autonomously and identifies locations where the robot will need to ask for help to successfully navigate to its destinations. In particular, CoBot-2 needs help pressing the up/down buttons outside the elevator, determining which elevator to enter, and holding the elevator door open before it can navigate into the elevator. Once inside the elevator, it needs help pressing the destination floor button and may sometimes need help determining when it is on the correct destination floor before leaving the elevator. CoBot-2 depends on building occupants who are also taking the elevator to be task helpers. These task helpers are already performing the actions themselves and therefore have low cost to help the robot as well.

A. Symbiotic Interaction with Task Solicitors

While other robots have asked for help from passers-by in the environment [9], [10], CoBot-2 must interact with task solicitors who requested the task and/or occupy the destination locations in order to manipulate objects and confirm task completion. We divide these interactions into those that happen at the *start* and at the *end* of a task.

Transport and Escort tasks both require *start* interactions. In order to transport an object or escort a person, CoBot-2 asks to acquire the object or find the person at the pickup location, saying "Hello, I'm here to take [object/person] to [room]. Press 'Done' when you are ready to go" and displays a button on the user interface. This interaction gives occupants time to find the object and arrange it on the robot or visitors time to finish conversations prior to being escorted. Additionally, the confirmation is a signal to begin navigating to the destination without having to actively sense each potential object or person.

At the *end* of each task, CoBot-2 also interacts with task solicitors. At the end of deliver message tasks, the robot asks "Hello. I have a message from [task solicitor]. Are you ready to hear it?". Then, after CoBot-2 speaks the message, it asks "Would you like me to repeat myself, or can I leave?" and displays a "Repeat" and a "Done" button. For all other tasks, CoBot-2 only confirms it has completed its task, saying "Please press 'Done when I can leave."

B. Waypoint Path Planning for Navigation

CoBot-2 also plans the autonomous and symbiotic interactions to navigate between rooms. Since CoBot-2 navigates in a multi-floor building with many occupants, it takes into account their preferences for traveling by their offices and the need to use the elevator. For each destination, the task executor calls the waypoint path planner to generate a low-cost path. The waypoint path planner keeps a room graph

$$G_R = \langle V_R, E_R \rangle,$$
$$V_R = \{v_i = (x_i, y_i)\}_{i=1:|V_R|},$$
$$E_R = \{e_i = (v_{i1}, v_{i2}) : v_{i1}, v_{i2} \in V_R\}_{i=1:|E_R|}.$$

The vertices $V_i \in V_R$ indicate the locations of offices and other important landmarks such as elevators and kitchens. The edges $e_i = (v_{i1}, v_{i2}) \in E$ indicate that the vertices v_{i1} and v_{i2} are connected by a navigable path. The cost function $c(e_i)$ of the edge is based on the length of the edge, the time to travel the edge, the capability of the robot to navigate there autonomously, and the cost to humans of traversing the edge. For example, the robot may lower the cost of an edge in order to favor particular edges if the building occupants in offices near that stretch of hallway enjoy watching CoBot-

2 go by. The waypoint path planner uses Dijkstra's shortest path algorithm with edge weights $c(e_i)$ and returns a path of waypoint locations

$$\vec{l_d} = \langle l_d^0 = v_{start}, l_d^1, l_d^2, \ldots, l_d^k = v_{end} \rangle$$

for the robot to navigate to get from v_{start} to v_{end}. Note that the path isn't necessarily the shortest distance path, but is rather the lowest cost path.

Given a set of waypoints, the path planner then classifies them into one of two categories: those to *direct* the robot towards more preferable paths and *stops* for help. If CoBot-2 determines that it is not capable of traversing an edge between two consecutive waypoint locations autonomously, it labels the first of the two waypoints in the path as a *stop*. There are three elevator stop locations:

- The location outside the elevator on the starting floor. CoBot-2 needs help pressing the up/down buttons and determining which elevator to enter.
- The location inside the elevator on the starting floor. CoBot-2 needs help pressing the floor number buttons inside the elevator.
- The location inside the elevator on the destination floor. CoBot-2 needs help identifying when it is on the destination floor and can exit the elevator.

All other waypoints in the path are *direct* to indicate that the robot can navigate between them autonomously.

C. Symbiotic Interaction with Task Helpers: Riding the Elevator

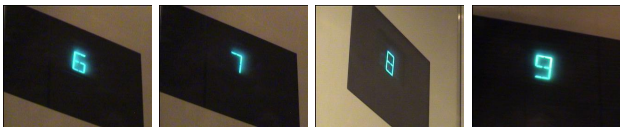
One of the most important contributions of the current successful deployment of CoBot-2 is the fact that the robot takes the elevator by itself, as our building has multiple floors. CoBot-2, in its symbiotic-autonomy depends on task helpers who are already using the elevator to help it too.

We view four robot states in riding the elevator, namely *waiting*, *entering*, *inside*, and *exiting* the elevator. Each of these states involves interaction with a human helper and corresponds to one of the three stops in the waypoint path planner's path. In the *waiting* state, the robot is outside of the elevator and asks to "Please push the up/down button" and to identify the arriving elevator as "Which elevator is going up/down" and displays the choices of two possible elevators (see elevators in Figure 1 behind the robot) on a touch screen. Given the human input, before the robot starts moving to position itself in front of the correct elevator, it requests "Please hold the elevator door." and then *enters* the elevator autonomously.

When *inside* the elevator, the robot can asks for help about its destination floor "Can you please push [the destination floor] button and tell me when we get to that floor." At the destination floor, the task executor expects a response to its request and at that time will wait for the doors to open and autonomously navigate out. If a person tells the robot incorrectly that it is on the destination floor, as soon as CoBot-2 exits the elevator, it can localize itself and autonomously detect that it is on the wrong floor. At that time, the task executor replans its path to the destination, asking for help to enter the elevator again.

We have also developed an approach to allow the robot to autonomously read the floor numbers displayed inside the elevator to identify when it is on the destination floor, rather than wait for a response to the request on the correct floor. The floor numbers are in a dedicated LED panel, which the robot can a) *detect* and b) *classify* using its vision camera. The detection algorithm uses the localization information of the robot inside the elevator, controls the camera pan to search for the known location of the LED number display in each elevator, tilts the camera as needed in its search, and zooms on the number. When the robot finds the LED number, it holds its camera to its position, and invokes the classifier. We trained an SVM, using a linear kernel and features given by histograms of gradient on the V channel of the HSV images. We trained one SVM per floor number using a one versus all approach. While the elevator is moving, CoBot-2 sees a small number of images (4-5) of each floor number. By using majority consensus over the last three images, CoBot-2 is able to identify the floor number with high accuracy without running the risk of missing floor transitions. Figure 3 shows examples of different elevator numbers and views obtained from the robot perspective. The numbers are zoomed, pre-processed, and classified.

(a) Robot view (b) Zoomed view (c) Pre-processing



(d) Six (e) Seven (f) Eight (g) Nine

Fig. 3. Elevator floor numbers as processed by CoBot-2.

The detection and classification algorithm falls back to the default behavior of asking a human for help if it reaches preset autonomous number reading thresholds. We tested this autonomous elevator number detection, and it is now being incorporated into the deployed CoBot-2.

IV. Localization and Navigation

Although CoBot-2 is symbiotic and relies on humans for help, it localizes and navigates fully autonomously.

A. Localization

CoBot-2 localizes with the Corrective Gradient Refinement (CGR) localization algorithm [11], using the laser range-finder and the plane filtered point cloud [12] from the Kinect sensor, along with wheel odometry. CGR localizes using a vector map M of the building extracted from the architectural plans. This map consists of a set of lines $l_i \in M$ representing the building walls.

CGR localization uses an observation model where points observed by the laser rangefinder and the Kinect sensor are associated with lines on the map, and the probability of the observation is computed as the joint probability of the observed points arising from the associated lines on the vector map. Let the set of 2D points observed by the laser rangefinder be denoted by $P = \{p_i\}_{i=1:n}$ and the pose of the robot by $x = \{x_l, x_\theta\}$ where x_l is the 2D location of the robot and x_θ its orientation angle. For every point $p_i \in P$, line $l_i \in M$ is found by ray casting such that the ray in the direction of $p_i - x_l$ and originating from x_l intersects l_i before any other line. The perpendicular distance d_i of p_i from the (extended) line l_i is computed. The total (non-normalized) observation likelihood $p(P|x)$ is then given by $p(P|x) = \prod_{i=1}^{n} \exp\left[-\frac{d_i^2}{2f\sigma^2}\right]$. Here, σ is the standard deviation of the distance measurements of a single ray, and f (where $f > 1$) is a discounting factor to discount for the correlation between rays. This observation model is also used to analytically compute the state space gradients of the observation model, which are used for the "refinement" step in CGR.

CGR localization using the Kinect depth camera [12] is based on the assumption that only large planar features observed in the Kinect depth image correspond to lines (walls) in the vector map. Using Fast Sampling Plane Filtering (FSPF) [12], the depth image is used to generate a "plane filtered point cloud" $P = \{p_i, r_i\}_{i=i:n}$ consisting of n points p_i and normals r_i corresponding to planes observed by the robot. Sampled points that did not fit the detected planes are added to the "outlier point cloud". The plane filtered points and their corresponding plane normal estimates are then projected into 2D to yield the set of points $P' = \{p'_i, r'_i\}_{i=1:n'}$ where p'_i are the projected 2D points, and r'_i the corresponding 2D normals. The projected points in P' are then used to compute the observation likelihood using the same observation model as is used for the laser rangefinder. The plane filtered points, as well as the outlier 3D points, are used to compute obstacle avoidance margins for the robot. CGR localization (using the laser rangefinder and Kinect sensors) thus provides the estimated pose (x, y, θ) of CoBot-2 on a particular floor of the building.

To detect when CoBot-2 has entered a new floor (using the elevator), we use the StarGazer sensor. The StarGazer sensor is an off-the-shelf sensor which has a ceiling-facing infrared camera and infrared LEDs. Retroreflective markers, consisting of patterns of dots, reflect the light from the infrared LEDs and are read by the infrared camera. The StarGazer sensor provides the unique identification code of the marker and the robot's pose relative to the marker. Thus, by placing unique StarGazer markers outside the elevator doors on every floor, CoBot-2 can immediately identify which floor it is on after exiting the elevator.

B. Navigation

CoBot-2 uses a graph based navigation planner [13] to plan paths between locations on the same floor of the building. The navigation graph G, distinct from G_R presented previously, is denoted by $G = \langle V, E \rangle$, $V = \{v_i = (x_i, y_i)\}_{i=:n_V}$, $E = \{e_i = (v_{i1}, v_{i2}) : v_{i1}, v_{i2} \in V\}_{i=:n_E}$. The set of vertices V consists of n_V vertices $v_i = (x_i, y_i)$ that represent the location of the ends and intersections of hallways. The set of edges E consists of n_E edges $e_i = (v_{i1}, v_{i2})$ that represent navigable paths between vertices v_{i1} and v_{i2}.

Given a destination location $l_d = (x_d, y_d, \theta_d)$, the navigation planner first finds the projected destination location $l'_d = (x'_d, y'_d, \theta_d)$ that lies on one of the edges in the graph. This projected destination location is then used to compute a topological policy using Dijkstra's algorithm for the entire graph. The navigation planner projects the current location $l = (x, y, \theta)$ onto the graph and then executes the topological policy until the robot reaches the edge on which l'_d lies, and then drives straight to the location l_d. Thus, the navigation planner navigates between start and end rooms $\in G_R$ given by the task executor, irrespective of whether or not they actually lie on the graph.

While executing the navigation plan, CoBot-2 performs obstacle avoidance based on the obstacles detected by the laser rangefinder and Kinect sensors. This is done by computing open path lengths available to the robot for different angular directions. Obstacle checks are performed using the 2D points detected by the laser rangefinder, and the down-projected points in the plane filtered and outlier point clouds generated by FSPF from the latest Kinect depth image.

V. Deployment Results

We deployed CoBot-2 on the upper four floors of an office building for a two week period. CoBot-2 was deployed for two hours every weekday and made available to the building occupants. Occupants were alerted of CoBot-2's availability through email and physical signs posted on bulletin boards and on the robot itself. The deployment times varied each day, and were announced beforehand on CoBot-2's website.

The response to CoBot-2's deployment was positive: over one hundred building occupants registered to use CoBot-2 on the website. Users found creative ways to exploit the robot's capabilities, including, but not limited to:

- Sending messages to friends.
- Reminding occupants of meetings.
- Escorting visitors between offices.
- Delivering printouts, inter-office mail, USB sticks, snacks, owed money, and beverages to other building occupants.

TABLE I
TOTAL NUMBER OF TASK REQUESTS PER TASK TYPE AND THE RESPECTIVE NUMBER THAT USED THE ELEVATOR.

Task Type	Total Requests	# Multi-floor
Escort	3	2
GoToRoom	52	22
DeliverMessage	56	20
Transport	29	22

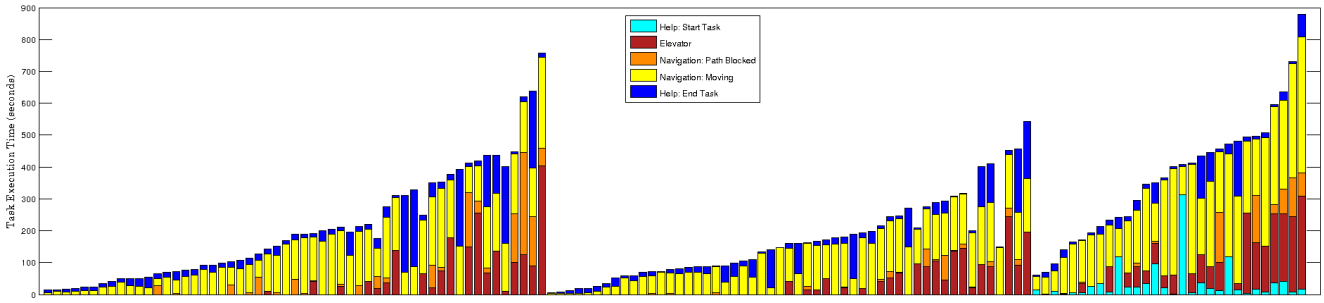


Fig. 4. Execution times for, from left to right, Deliver Message tasks, Go to Room tasks, and Transport tasks. The breakdown includes 1) waiting for help to start the task, 2) riding the elevator, 3) navigating (not including time blocked by obstacles), 4) waiting blocked by an obstacle, and 5) waiting for help to end the task.

Particularly in the first couple days of deployment, we found building occupants following the robot around to see where it was going and how it worked.

We found that occupants scheduled the robot to transport objects between multiple floors of the building more often than they used the multi-floor functionality for other tasks (see Table I). In particular, the transport task saved the task solicitors time because they did not have to travel between floors themselves. However, even the other scheduled tasks utilized the elevator 40% of the time.

In fulfillment of the user requested tasks, CoBot-2 travelled a total of 8.7 km, which covered most of the building. CoBot-2 spent

- 6 hours and 17 minutes navigating this distance,
- 36 minutes with a blocked path waiting for a person to move out of its way,
- 1 hour and 2 minutes waiting for help with the elevator,
- 1 hour and 18 minutes waiting for task solicitor help to complete its tasks.

Figure 4 shows how much time CoBot-2 took to execute each task, and how that time was apportioned. A total of 140 tasks were completed during the two week deployment, which took 9 hours and 13 minutes. Based on these times, we find that task solicitors quickly responded to the robot's request for help at the start and end of tasks. Building occupants (even those that had never scheduled a task) were willing and able to help the robot in and out of the elevator. This finding supports our model of symbiotic autonomy— humans are willing to help a robot complete its tasks so that the robot is available and capable of performing tasks for them as well.

Although CoBot-2 could be required to wait for human help indefinitely, the task execution times are limited. In general, little more than five minutes per task was spent in the elevator. This is because if CoBot-2 spends more than five minutes waiting for human help, it sends an email to our research group asking for assistance. Typically, however, occupants helped CoBot-2 and there was no need to do so. Furthermore, if at the end of a task no human pressed the button to indicate that the task was complete, CoBot-2 marked the task as complete and moved on to the next task.

VI. Conclusion

We have successfully deployed an autonomous robot to service users in a multi-floor building. We have presented how users request tasks dynamically on a web server, and how requests are scheduled via a mixed integer program. Our task planner and executor divides each task into autonomous navigation actions, and interactions with humans which require help. Our robots are symbiotic, helping occupants of the building by fulfilling their requests, while also receiving help from humans to complete tasks and to use the elevators. The robots are also fully autonomous; they navigate and localize on their own. CoBot-2 has traveled over 100km throughout its lifetime, and we present results of 8.7km for public deployment.

References

[1] N. J. Nilsson, "Shakey the robot," SRI International, Tech. Rep. 323, 1984.

[2] R. Simmons, R. Goodwin, K. Z. Haigh, S. Koenig, and J. O'Sullivan, "A layered architecture for office delivery robots," in *Proceedings of the first international conference on Autonomous agents (AGENTS'97)*, 1997, pp. 245–252.

[3] W. Burgard, A. B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun, "Experiences with an interactive museum tour-guide robot," *Artificial Intelligence*, vol. 114, pp. 3–55, October 1999.

[4] T. Fong, I. Nourbakhsh, and K. Dautenhahn, "A survey of socially interactive robots," *Robotics and Autonomous Systems*, vol. 42, pp. 143–166, 2003.

[5] S. Thrun *et al.*, "Probabilistic algorithms and the interactive museum tour-guide robot minerva," *The International Journal of Robotics Research*, vol. 19, no. 11, pp. 972–999, 2000.

[6] U. Visser and H.-D. Burkhard, "RoboCup: 10 years of achievements and future challenges," *AI Magazine*, vol. 28, no. 2, pp. 115–132, Summer 2007.

[7] S. Rosenthal, J. Biswas, and M. Veloso, "An effective personal mobile robot agent through a symbiotic human-robot interaction," in *International Joint Conference on Autonomous Agents and Multiagent Systems*, 2010, pp. 915–922.

[8] A. Larsen and O. Madsen, "The dynamic vehicle routing problem," Ph.D. dissertation, Technical University of Denmark, Department of Informatics and Mathematical Modeling, 2000.

[9] H. Hüttenrauch and S. Eklundh, "To help or not to help a service robot: Bystander intervention as a resource in human-robot collaboration," *Interaction Studies*, vol. 7, no. 3, pp. 455–477, 2006.

[10] A. Weiss, J. Igelsböck, M. Tscheligi, A. Bauer, K. Kühnlenz, D. Wollherr, and M. Buss, "Robots asking for directions: the willingness of passers-by to support robots," in *HRI '10*, 2010, pp. 23–30.

[11] J. Biswas, B. Coltin, and M. Veloso, "Corrective gradient refinement for mobile robot localization," in *Intelligent Robots and Systems (IROS), 2011 IEEE International Conference on*. IEEE, 2011.

[12] J. Biswas and M. Veloso, "Depth camera based indoor mobile robot localization and navigation," in *2012 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2012.

[13] B. Coltin, J. Biswas, D. Pomerleau, and M. Veloso, "Effective semi-autonomous telepresence," *Proceedings of the RoboCup Symposium*, pp. 289–300, July 2011.

Decoupling Behavior, Control, and Perception in Affordance-Based Manipulation

Tucker Hermans James M. Rehg Aaron F. Bobick

***Abstract*— A novel mechanism is introduced by which a robot can connect the general notion of an affordance of an object to specific behaviors by which the robot can achieve the desired action. We achieve this by decomposing an affordance-drive behavior into three components. We first define *controllers* that specify how to achieve a desired change in object state through changes in the agent's state. For each controller we develop at least one *behavior primitive* that determines how the controller outputs translate to specific movements of the agent. Additionally we provide at least one *perceptual proxy* that defines the representation of the object that is to be computed as input to the controller during execution. A variety of proxies may be selected for a given controller and a given proxy may provide input more than one controller. Decoupling these components allows the systematic exploration of a variety of strategies when evaluating the affordances of novel objects. We demonstrate the approach using a PR2 robot that executes different combinations of controller, behavior primitive, and proxy to perform a push positioning behavior on a selection of household objects.**

I. Introduction

As the goal of having robots operate in uncontrolled environments becomes more critical to the advancement of robotics, there has been much research on the notion of *affordances* of objects with respect to a robot agent [1]. Within the context of robotics affordances describe the possible actions an agent can take acting upon an object and the resulting outcome [2]. Specific examples might include *graspable* (e.g. [3]) or *pushable* [4] that indicate a particular object can be grasped or pushed, respectively. Because one can cast affordances as state-action pairs that will transform the object state in some way, there has been further work in considering affordance as a basis of planning [5]. If the robot has a goal of clearing the path to an object being fetched, it might first push interfering objects to the side assuming they can be pushed, i.e. have the affordance *pushable*.

However, while a planner may be able to leverage an abstracted description of the affordance as being true or not of an object, or even of having some probability of being true in the case of a probabilistic planner, such a high level description is not sufficient to actually *execute* the action required for the affordance. And, indeed the method of performing the action may vary by object or object state: pushing a round cereal bowl might be quite different than pushing a TV remote control that has rubber buttons that occasionally stick to a table surface.

Tucker Hermans, James M. Rehg, and Aaron F. Bobick are with the Center for Robotics and Intelligent Machines and The School of Interactive Computing, Georgia Institute of Technology, Atlanta, GA. {thermans, rehg, afb}@cc.gatech.edu

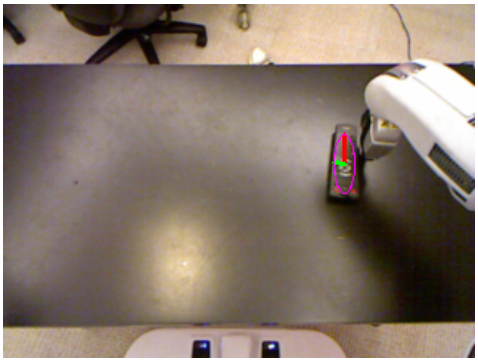


Fig. 1: Example of the robot performing pushing using feedback from visual tracking. The red line represents the dominant orientation of the object computed from the purple ellipse fit to the object's 3D point cloud.

The goal of this paper is to introduce a mechanism by which a robot can attach the general notion of an affordance to a specific method by which the robot can achieve it. We do this by decomposing an affordance-driven behavior into three components. We first define *controllers* that specify how to achieve a desired change in object state through changes in the agent's state. For each controller we develop at least one *behavior primitive* that determines how the controller outputs translate to specific movements of the agent. Additionally we provide at least one *perceptual proxy* that defines the representation of the object that is to be computed as input to the controller during execution. Obviously, the proxy must be sufficiently rich to support estimation of the variables required by the controller. The novelty here is that multiple proxies may support the same controller and a given proxy representation may be selected for use with more than one controller. Additionally, a single behavior primitive may be compatible with multiple controllers. Decoupling these components allows the systematic exploration of a variety of strategies when evaluating the affordances of novel objects.

In this paper we use as an example affordance *push-positionable* where the goal is to move the object to a specified location. We develop two feedback controllers to implement this action using the *overhead push* behavior primitive. Each of these controllers has its own perceptual proxy. These methods require no prior knowledge of the object being pushed and make no estimates of underlying model parameters. We show how a particular controller may succeed or fail on the basis of the proxy computed. Finally, we show how one of these controller-proxy pairs can be utilized by a second behavior primitive, a *sweep push*.

We organize the remainder of our paper as follows. Section II describes relevant past work on the topic of affordance learning and affordance-based planning; we also briefly mention prior methods of controller-based pushing. In Section III we formally define the affordance assertion problem and the push-positioning task. Then in Section IV we present two proposed feedback controllers each based upon a different perceptual proxy and each better suited to different object types. We give details of our implemented proxies in Section V followed by the implemented behavior primitives in Section VI. Section VII presents results of experiments performed on a robot using our proposed system. We conclude with directions for future work in Section VIII.

II. Related Work

In early work on affordance prediction described in [6, 7], a humanoid robot learns to segment objects through actions such as poking and prodding. After interaction with a set of objects, the system could learn the rollable affordance for the objects and predict the result of hand-object interactions. The goal was to learn parameters such as initial location of the hand with respect to the orientation of an object that best induce the desired motion. The actions were atomic in the sense that they were applied in their entirety and the results measured. In [8], a classification method is applied to high-level image features to learn the affordance of liftable. Using decision tree classifiers with SIFT and patch features, they demonstrate the ability to learn liftable vs nonliftable objects.

A series of works [9–11] address the task of recognizing the graspable and tappable affordances, based upon experimentation through self-observation of actions. Learning in a Bayesian network is employed to learn cuing rules for actions. The network models the relationship between object appearance and motion, end-effector motion, and action. In [12], a functional approach to affordance learning is developed in which subcategories of the graspable affordance (such as handle-graspable and sidewall-graspable) are learned by observation of human-object interactions. Interaction with specific object parts leads to the development of detectors for specific affordance cues (such as handles). The focus of that work was to learn a mapping from object features to grasp locations without unduly worrying about what method of grasping would work at that location.

Related, Stoytchev [13] describes a method for learning the functionality of a tool through observation of the effects of exploratory behaviors, a process that he termed behavioral babbling. In experiments with a mobile manipulator, the system demonstrated the ability to learn the affordances of a set of tools that could be identified by their color.

With respect to planning, affordance-based modeling of robot-object interaction would allow a planning system to systematically select from a set of actions to achieve desired subgoals. An example of such an approach is given in [5] where the robot arrange plates and bowls on a table. In that work, however, there is an assumption of a priori knowledge as to which behaviors can successfully operate on which objects and what the resulting state of the action will be. The approach presented here would both permit experimental exploration on the part of the robot of the different methods by which an affordance could be realized for a given object and a method for monitoring the effectiveness of the behaviors.

The concept of Instantiated State Transition Fragment (ISTF) is introduced in [14]. It encodes the pairing between an object and an action in the context of the state transition function for a domain-specific planner. The authors describe a process of learning Object Action Complexes (OACs) through generalization over ISTF's. Montesano et. al. [11] present a Bayesian network model that implicitly represents affordances as mappings from action to effect, which are mediated by the visual features of objects. A model for grasping, tapping, and touching actions is learned from both self-observation and imitation of a human teacher. The goal is to leverage such OACs in planning and executing a multi-step task.

Effective pushing behaviors offers a number of benefits in robotics domains which complement standard pick-and-place operations. For example pushing can be used to move objects too large for the robot to grasp, to more quickly move objects to new locations, or to move an object while another object is already grasped. As such there has been considerable work at developing such capability. Early work that analyzed a complete model of the dynamics of pushing was developed by Mason who describes the qualitative rotational changes of sliding rigid objects being pushed by either a single point or single line contact [15]; representative examples of some more recent applications of pushing are available in [4, 16–20].

Notably, Ruiz-Ugalde et al. execute a pushing behavior by determining the static and kinetic friction coefficients for multiple objects with rectangular footprints, both between the robot hand and object and between the object and table [20]. Additionally they present a robust controller using a cart model for the object being pushed. The control takes object velocity as input to control the system to a desired 2D pose, as such the mapping from applied force to velocity is believed known from the estimation and is separate from the control of the object. Their control is the closest approach we have found to the pushing controllers presented in this work. However, their overall approach presumes the ability to predict the resulting action based upon known or learned parameters that characterize the physics of the object.

To address the inherent difficulty in estimating model parameters, there are data-driven methods that use an empirically derived characterization of the outcomes of specific actions applied to the object. For example, Narasimhan uses vision to determine the pose of polygonal objects of known shape in the plane [21]. Three methods were proposed to be able to push objects into the desired location and orientation: a hand coded heuristic that assumes known center of mass (and uniform friction properties), a feedback controller to explicitly rotate and translate an object, and finally a data-driven, learning approach that stores the results of different pushes and uses nearest neighbor to select the action that generates a result closest to the desired outcome. where the

Fig. 2: Initial pose of the food box. The green circle represents the desired position and the green line is the current vector between the object origin and the goal.

states and results of different methods are examined.

Similarly, Salganicoff et al. present a method for learning and controlling the position in image space of a planar object pushed with a single point contact [22]. Slip of the object is avoided by pushing at a notch in the object. Scholz and Stilman learn object specific dynamics models for a set of object through experience [23]. Each object is pushed at a number of predefined points on the perimeter and the robot learns Gaussian models of displacement in (x, y, θ) at each location. These learned models are then used to select the input push location given a desired object pose.

III. Problem Statement

We define an affordance to exist between a robot and an object, if the robot can select a specific behavior primitive, controller, and perceptual proxy by which it can successfully perform the desired action. We take as an example action that of *push positioning*, where the robot must position an object at an arbitrary location by pushing with its arm. We assume that the object is being pushed over a plane and thus the object state $X = (x, y)$ defines the location of the origin of the object in a 2D space.[1] We denote the goal pose as $X^* = (x^*, y^*)$. This state representation is sufficient at the level of a task level planner, however, a specific controller may require more state variables to be estimated by the relevant perceptual proxy.

The (unknown) dynamics of the pushing system are governed by the nonlinear relation $\dot{X} = h(X, Q, U)$ which defines the interaction dynamics between the object state, the robot configuration Q, and the input to the robot U. Importantly, we make no attempt model h. In developing our visual feedback controllers to achieve the above defined task, we presume we do not have an exact measurement of the object state. Instead we will operate on the estimated state $\hat{X}$ that will be computed at each timestep based upon properties of a perceptual proxy. In this work we control the arm through Cartesian control, both position and velocity, in the robot's task frame. We denote the specific forms of U and X used in our controllers in detail below. Our task thus becomes defining a feedback control law $U = g(\hat{X}, X^*)$ which drives the position error $X_{err} = X^* - \hat{X}$ to zero.

[1]We wish to make clear, that we do not assume objects are flat.

IV. Two Push-Positioning Controllers

In this section we define two visual feedback controller for the robot to push an object to a desired location. Each controller has a necessary set of state variables to be estimated from the perceptual representation that is continuously updated. These representations serve as the proxies for the object with respect to the defined controllers.

A. Spin-Correction Control

Our first method of defining a push-positioning controller relies on the fact that the direction of an object's rotation while being pushed depends on which side of the center of rotation the applied force intersects. This fact is well described by the limit surface formulation [15, 24]. Mason derived the velocity direction of a sliding object as a function of the forces applied by the pushing robot as well as the support locations and mass distribution of the object [15]. These parameters are difficult to know or estimate well for a given object and even when they are known, the exact resulting behavior is often indeterminate [15]. However, we make use of Mason's realization that the resulting rotation of the object abruptly changes direction when the input force passes directly through the center of rotation of the object. As such we can use the direction of the observed rotation of the object to infer which side of the center of rotation the applied forces are currently acting through. We can then correct the direction of our applied forces to compensate for any unwanted rotation of the object.

Since objects tend to rotate less when the input forces as directed near the center of the object our controller attempts to push the object through its center in the direction of the goal position. This gives a simple procedure for determining the initial hand position. We cast a ray from the goal location through the centroid of the object and find its intersection with the far side of the object. This location defines the initial position for the hand. We further orient the hand so that its gripper is facing in the direction of the goal from the initial position. An example image of the initial hand placement can be seen in Figure 2. Once positioned our feedback control process is initiated. The controller is defined in equations 1 and 2 which operates on state $X = (x, y, \theta, \dot{\theta})$ and computes input U of x and y velocity of the end effector in the robot's workspace.

$$u_{\dot{x}} = k_g v_{goal_x} - sin(\phi_g)(v_{rot}) \quad (1)$$

$$u_{\dot{y}} = k_g v_{goal_y} + cos(\phi_g)(v_{rot}) \quad (2)$$

Our control is comprised of two terms. The first pushes through the object driving it to the desired goal, while the second displaces the contact location between the robot and object to compensate for changes in object orientation. The input control defined in equations 3 and 4 commands the robot to push in the direction of the goal. The overall effect of this component is controlled by the positive gain k_g. Since the object lies between the end effector and the goal this

causes the object to translate towards the goal.

$$v_{goal_x} = (x^* - \hat{x}) \quad (3)$$

$$v_{goal_y} = (y^* - \hat{y}) \quad (4)$$

However, since the forces applied by the robot on the object are not pushing directly through the center of rotation, the object will undoubtedly spin. To compensate for this we apply additional input velocities proportional to the observed rotational velocity of the object. We desire not only that the object not rotate, but also that it maintains its initial orientation θ_0. We combine these terms to generate v_{rot}.

$$v_{rot} = k_{sd}\dot{\hat{\theta}} - k_{sp}(\theta_0 - \hat{\theta}). \quad (5)$$

We desire to displace the end effector perpendicular to the current direction of the object's translational motion. Since our estimate of the instantaneous velocity is somewhat noisy, we instead rotate the velocity vector about the angle defined between the center of the object and the goal ϕ_g.

$$\phi_g = \texttt{atan2}(y^* - \hat{y}, x^* - \hat{x}) \quad (6)$$

Our pushing controllers halt once $x_{err} < \epsilon_x$ and $y_{err} < \epsilon_y$. For the purpose of developing this method as well as the controller in Section IV-B, the gains are manually adjusted, but remain fixed for all objects.

B. Centroid Alignment Control

Our second push-positioning controller replaces the monitoring of object orientation with a strategy based upon the relative locations of the object's centroid, the assumed location of the contact point on the end effector, and the goal position. The simple intuition is that pushing the object can be achieved by positioning the end effector at a location on the object boundary that intersects a line between the goal location and the object centroid.

The robot achieves this behavior by using a control law that includes a velocity term to move toward the goal and one that moves the end effector to the line defined through the goal centroid locations:

$$u_{\dot{x}} = k_{gc}v_{goal_x} + k_c v_{centroid_x} \quad (7)$$

$$u_{\dot{y}} = k_{gc}v_{goal_y} + k_c v_{centroid_y} \quad (8)$$

where v_{goal_x} and v_{goal_y} are as before. The second term provides the additional velocity term toward the goal-centroid line; $v_{centroid_x}$ and $v_{centroid_y}$ are components of perpendicular vector from the presumed end effector contact point to the goal-centroid line. The robot then pushes in the direction of the goal attempting to maintain this collinearity relation. This controller has the state $X = (x, y)$ and computes the same U as in Section IV-A. Additionally, the end effector is initially positioned relative to the object as above.

V. Object Proxies

The two above controllers have modest perceptual requirements. The orientation-velocity controller requires both the location of the object and its orientation whereas the centroid-driven one only requires position as defined by the object centroid. Here we describe the perceptual computations performed and the proxies that satisfy the requirements.

Fig. 3: The first image shows the overhead push behavior primitive pushing the television remote. The second image show the sweep push behavior primitive pushing the dinner bowl. Both objects have the estimated centroid location and ellipse overlayed.

We begin with a simple depth-based segmentation and tracking method that currently assumes only a single object resting on the sliding surface (a table) is in the scene. The input is the RGBD image of a Microsoft Kinect though in this simple implementation only the depth channel is used. We initialize the tracker by moving the robot's arms out of the view of the camera, capture the depth image and then use RANSAC [25] to find the dominant plane in the scene parallel to the ground plane. We then remove all points below the estimated table plane and cluster the remaining points. We filter out clusters with very few points and, because we're assuming only one object is on the table, we accept the cluster with most points as the object.[2] We compute the 3D centroid of the points in the cluster and use the x and y components as the object's location on the table.

Once initialized we track by performing the same procedure with the added step of removing points belonging to the robot from the scene. We project the robot model into the image frame using the forward kinematics of the robot and remove points from the point cloud coincident with the robot arm mask. Because of noise in measurements and other calibration issues points belonging to the robot can sometimes remain. To prevent the tracker from selecting any of these points as the current object we perform nearest neighbor matching between current cluster centroids and the previous object state, selecting the closest as the current object. We then estimate the object velocity using the previous estimate of the object state.

Computing the perceptual proxies needed for each of the controllers is straightforward given the tracker described above. For the centroid based control where the proxy is only the centroid of the object, we can immediately return the x and y values. For the orientation-velocity control we need a proxy that includes an estimate of object orientation, as well as its rotational velocity, with respect to the global robot frame. We fit a 2D ellipse [27] to the x and y values of all points in the object point cloud and use the orientation of the major axis of the ellipse as the objects orientation θ. The change in θ from one frame to the next is the

[2]We note that we [19] and others (e.g. [26]) have previously developed methods for singulating objects form each other by pushing actions.

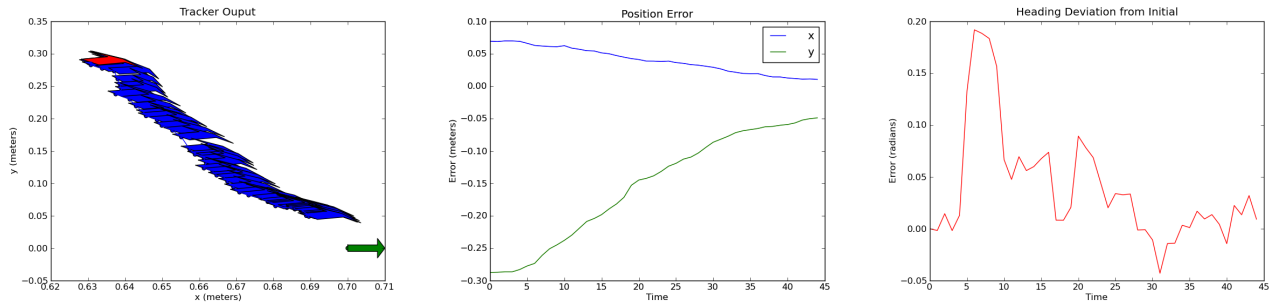


Fig. 4: The first image shows the tracked box pose trajectory. The red error shows the initial pose. The green arrow is the goal pose. The second image is error in the food box position and the third shows change in orientation from the initial orientation.

estimated orientation velocity $\dot{\theta}$. An example of the computed ellipsoidal proxy is shown in Figure 1.

Note that these two simple proxies — the first merely a centroid, the other the 2D ellipse — are intended to be available for *any controller* for which these inputs are sufficient. And, indeed a robot may have a variety of proxies that can yield a set of controller input variables. Later, when we discuss future work of learning the affordances of objects, we will return to this point.

VI. Pushing Behavior Primitives

We performed pushing with two behavior primitives: an overhead push and a sweep push. The overhead push has the robot place its hand such that the fingertips are in contact with the table with the wrist directly above. The sweep push places the length of the hand on the table with the flat of the hand facing the object. As our controllers operate only within the 2D pose of the hand (x, y, θ), the configuration of the end effector with respect to the arm and object remain fixed during operation. Specifically that means that the wrist remains above the hand throughout pushing for the overhead push. Likewise the sweep push keeps the long side of the robot hand along the table with the broad side of the hand perpendicular to the surface during pushing. Images of the robot operating with these behavior primitives can be seen in Figure 3.

For both primitives the arm is moved to the initial pushing pose using Cartesian position control. The arm is first moved to a position directly above the table at the desired pushing pose and desired orientation. The hand is then lowered in a straight line to the initial pushing pose. We use a Jacobian inverse controller to control the Cartesian velocity of the end effector during feedback control. We push objects with the robots right arm when the angle towards the goal pose move left in the workspace and use the left arm in the opposite case.

VII. Experimental Validation

We implemented our system on a Willow Garage PR2 robot augmented with a Microsoft Kinect for visual input. We experiment with different combinations of proxies, control laws, and behavior primitives in pushing a television remote, a food box, and a dinner bowl. In all experiments $\epsilon_x = \epsilon_y = 0.05$ meters.

(a) (b)

Fig. 5: The top and bottom of the television remote. The support distribution of the remote is much more complex than a simple polygon. Additionally the narrow end is significantly more massive than the wider end owing to the batteries inside.

A. Goal Position Controller Evaluation

We first show an example of pushing a television remote using the overhead push controlled by the spin correction controller. The perceptual proxy used is the ellipse model. The TV remote has a rather complicated set of support points and far from uniform mass or friction distributions. We show an up close picture of the remote in Figure 5. The tracked trajectory of the TV remote as well as the pose errors are shown in Figure 6. Midway through the pushing trajectory the remote becomes partially occluded by the robot arm which causes a jump in the estimated position. Figure 7 shows the controller compensating for this change which induces a larger velocity in the object, including its rotation. We show the velocities for the remote in Figure 8. Note that after the increased rotational velocity the controller most apply larger input velocities to maintain the initial orientation and continue pushing towards the goal. Regardless, the remote control converges within the desired bounds of the goal pose and the execution is successful.

We now show that this same affordance instantiation of overhead push, ellipse proxy, and spin correction controller

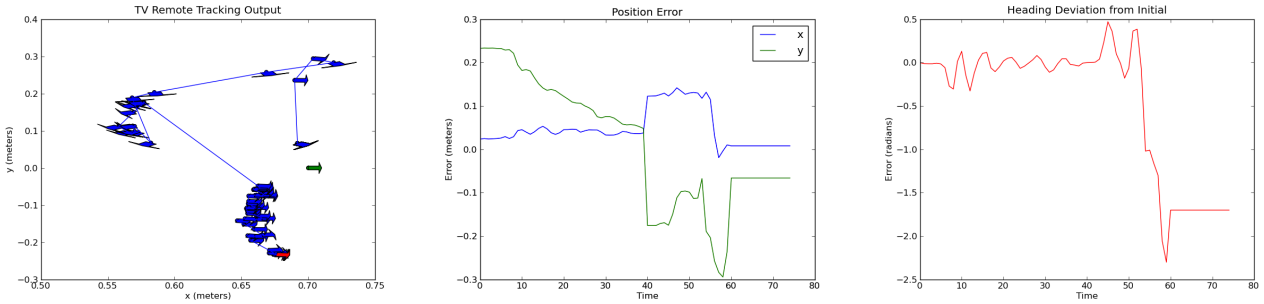


Fig. 6: The first image shows the tracked TV remote pose trajectory. The red error shows the initial pose. The green arrow is the goal pose. The large jump in error near time 40 is a result of the TV remote becoming partially occluded by the robot arm, which results in poor visual tracking performance.

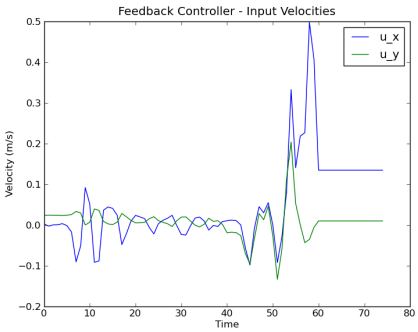


Fig. 7: Plot of the input velocities to the arm controller commanded by our feedback controller during pushing of the TV remote.

can be used for a different object of a simple food box. The robot successfully pushed this box using the same components as with the television remote. The results for a trial with this setting are shown in Figure 4. The box can be seen in Figure 2.

We investigate the same manipulation settings of overhead push, ellipse proxy, and spin correction controller to a simple, white dinner bowl. We show the robot pushing this bowl in Figure 3. The position error for the bowl and the input velocities during control are shown in Figure 9. We show the tracker output and rotational velocity estimates of the bowl in Figure 10. Applying this method to the bowl fails to push the bowl to the desired location. This failure can be attributed to the symmetric appearance of the bowl, which causes instability in estimating the object's orientation by the ellipse perceptual proxy. However, by pushing the bowl with the overhead push controlled by the centroid controller the robot can correctly position the object. We show error results and input velocities for these settings in Figure 11.

Following the success of the centroid controller in pushing the bowl, we investigate its use with the overhead pushing behavior primitive to push the television remote. Unsurprisingly, the centroid controller quickly loses contact with the remote since the visually estimated centroid is not the center of rotation and trying to push in line with it fails to compensate for the object's rotation. Figure 12 shows position errors and input velocities from the experiments.

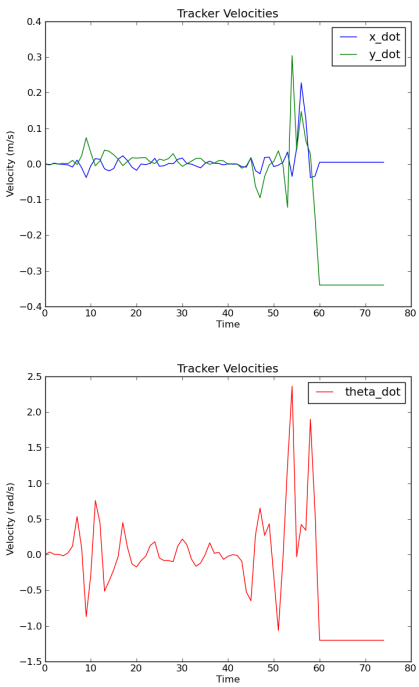


Fig. 8: Plot of the tracked object velocities of the TV remote.

B. Behavior Primitive Evaluation

We now examine using the sweep push behavior primitive with the controller proxy pairs. Following the success of positioning the TV remote with the spin correction controller and overhead push, we tried the same setup with the sweep push behavior primitive. This performed quite poorly. Partially at fault was the occlusion of the remote by the arm causing unstable state estimates. Additionally the spin compensating control input, v_{rot} caused somewhat volatile control of the sweeping end effector, that was much smoother with the overhead push. This could have perhaps been fixed by changing controller gains, however, we did not investigate this.

We also examined pushing the bowl using the centroid controller and the sweep push. This method successfully positioned the bowl. We show the position error and input velocities in Figure 13. The error results and control velocities were quite similar to those seen in pushing with the

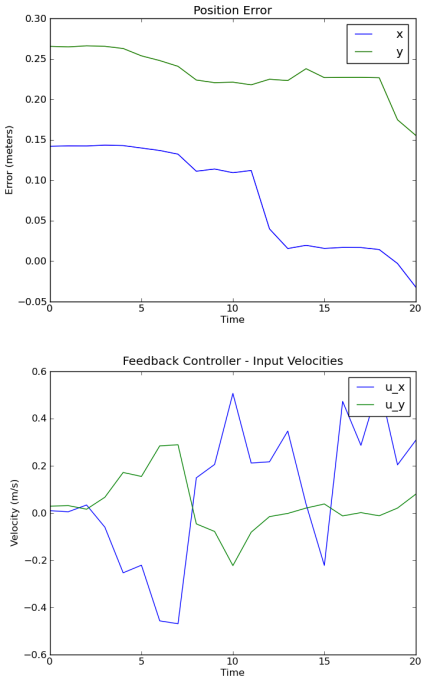


Fig. 9: Position error and input velocities for pushing the bowl using the spin correction controller with the overhead push.

overhead push.

VIII. Conclusions and Future Work

We have presented a novel mechanism by which a robot can connect the general notion of the affordances of an object to specific methods by which the robot can perform the necessary actions. We decompose affordance actions into behavior primitives, controllers, and perceptual proxies. This not only simplifies developing these capabilities but also allows a robot to systematically explore the affordances of objects. In the near future we will incorporate this approach into a learning paradigm where a robot not only learns the affordance of novel objects but also attempts to learn perceptual markers that will permit transfer of affordance knowledge between objects.

References

[1] J. J. Gibson, "The theory of affordances," in *Perceiving, Acting, and Knowing: Toward an Ecological Psychology*, R. Shaw and J. Bransford, Eds. Hillsdale, NJ: Lawrence Erlbaum, 1977, pp. 67–82.

[2] F. Iida, R. Pfeifer, and L. Steels, *Embodied Artificial Intelligence International Seminar, Dagstuhl Castle, Germany, July 7-11, 2003, Revised Papers*, ser. Lecture Notes in Computer Science. Springer, 2004, vol. 3139.

[3] A. N. Erkan, O. Kroemer, R. Detry, Y. Altun, J. Piater, and J. Peters, "Learning probabilistic discriminative models of grasp affordances under limited supervision," *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1586–1591, Oct. 2010. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5650088

[4] D. Katz, Y. Pyuro, and O. Brock, "Learning to Manipulate Articulated Objects in Unstructured Environments Using a Grounded Relational Representation," in *Proceedings of Robotics: Science and Systems IV*, Zurich, Switzerland, June 2008, pp. 254–261.

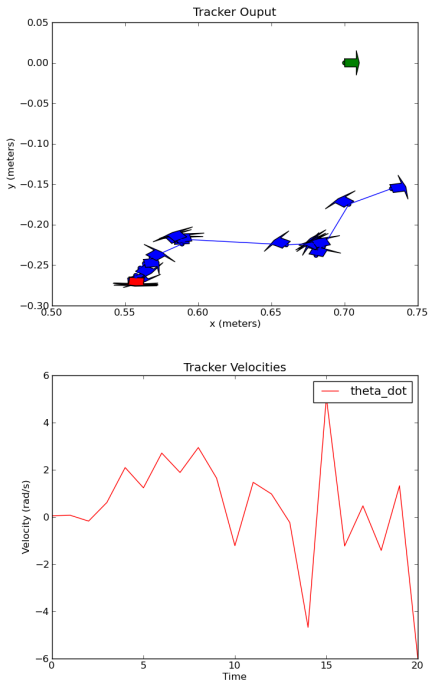


Fig. 10: Position and orientation estimates as well as rotational velocities estimates for pushing the bowl using the spin correction controller with the overhead push.

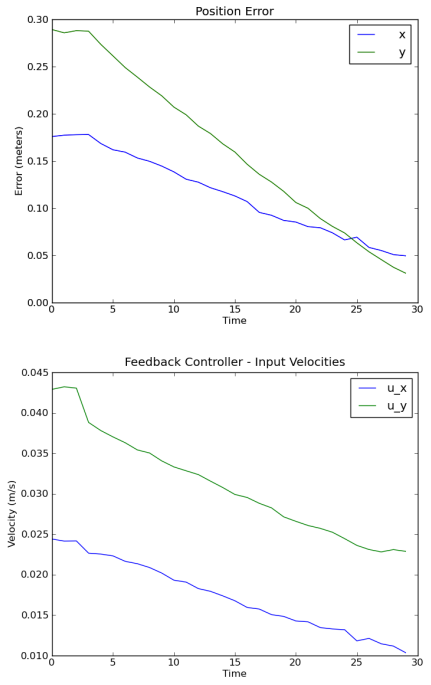


Fig. 11: Position error and input velocities for pushing the bowl using the centroid controller with the overhead push.

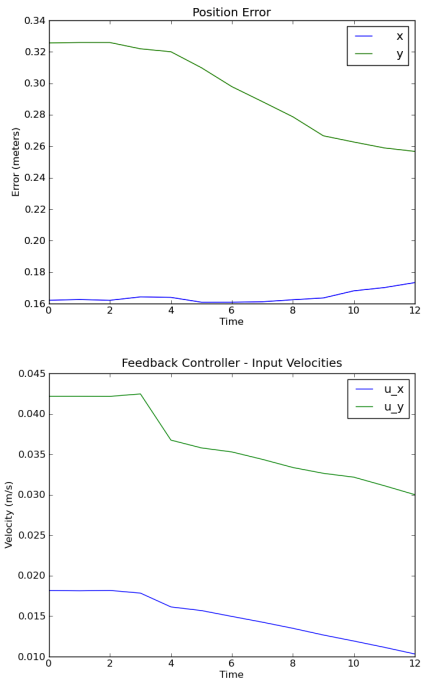


Fig. 12: Position error and input velocities for pushing the television remote using the centroid controller with the overhead push.

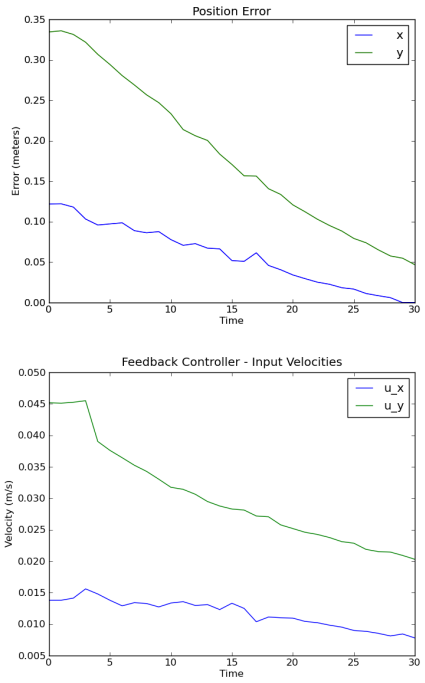


Fig. 13: Position error and input velocities for pushing the bowl using the centroid controller with the sweep push.

[5] J. Barry, K. Hsiao, L. P. Kaelbling, and T. Lozano-Perez, "Manipulation with multiple action types," in *International Symposium on Experimental Robotics*, no. 1122374, 2012, pp. 1–15.

[6] P. Fitzpatrick, G. Metta, L. Natale, S. Rao, and G. Sandini, "Learning about objects through action - initial steps towards artificial cognition," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, vol. 3, Sept 2003, pp. 3140–3145.

[7] G. Metta and P. Fitzpatrick, "Early integration of vision and manipulation," *Adaptive Behavior*, vol. 11, no. 2, pp. 109–128, 2003, special Issue on Epigenetic Robotics.

[8] G. Fritz, L. Paletta, R. Breithaupt, E. Rome, and G. Dorffner, "Learning predictive features in affordance based robotic perception systems," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Beijing, China, Oct 2006, pp. 3642–3647.

[9] M. Lopes, F. S. Melo, and L. Montesano, "Affordance-based imitation learning in robots," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, San Diego, CA, 2007, pp. 1015–1021.

[10] L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor, "Modeling object affordances using bayesian networks," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2007.

[11] ——, "Learning object affordances: From sensory-motor coordination to imitation," *IEEE Trans. on Robotics*, vol. 24, no. 1, pp. 15–26, Feb 2008.

[12] M. Stark, P. Lies, M. Zillich, J. Wyatt, and B. Schiele, "Functional object class detection based on learned affordance cues," in *Sixth Intl. Conf. on Computer Vision Systems (ICVS 08)*, Santorini, Greece, May 2008, pp. 435–444.

[13] A. Stoytchev, "Behavior-grounded representation of tool affordances," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2005.

[14] C. Geib, K. Mourao, R. Petrick, N. Pugeault, M. Steedman, N. Krueger, and F. W org otter, "Object Action Complexes as an Interface for Planning and Robot Control," in *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, Genova, Italy, Dec 4-6 2006.

[15] M. T. Mason, "Mechanics and Planning of Manipulator Pushing Operations," *The International Journal of Robotics Research (IJRR)*, vol. 5, pp. 53–71, September 1986.

[16] D. Omrčen, C. Böge, T. Asfour, A. Ude, and R. Dillmann, "Autonomous Acquisition of Pushing Actions to Support Object Grasping with a Humanoid Robot," in *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, Paris, France, 2009.

[17] M. Dogar and S. Srinivasa, "Push-Grasping with Dexterous Hands: Mechanics and a Method," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robotics and Systems (IROS)*, 2010.

[18] A. Cosgun, T. Hermans, V. Emeli, and M. Stilman, "Push Planning for Object Placement on Cluttered Table Surfaces," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robotics and Systems (IROS)*, 2011.

[19] T. Hermans, J. M. Rehg, and A. Bobick, "Guided Pushing for Object Singulation," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robotics and Systems (IROS)*, 2012.

[20] F. Ruiz-Ugalde, G. Cheng, and M. Beetz, "Fast Adaptation for Effect-aware Pushing," in *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, 2011.

[21] S. Narasimhan, "Task Level Strategies for Robots," Ph.D. dissertation, Massachusetts Institute of Technology, 1994.

[22] M. Salganicoff, G. Metta, A. Oddera, and G. Sandini, "A vision-based learning method for pushing manipulation," in *AAAI Fall Symposium on Machine Learning in Computer Vision*, 1993.

[23] J. Scholz and M. Stilman, "Combining Motion Planning and Optimization for Flexible Robot Manipulation," in *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, 2010.

[24] S. Goyal, A. Ruina, and J. Papadopoulos, "Limit Surface and Moment Function Descriptions of Planar Sliding," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 1989.

[25] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, pp. 381–395, June 1981. [Online]. Available: http://doi.acm.org/10.1145/358669.358692

[26] L. Y. Chang, J. R. Smith, and D. Fox, "Interactive Singulation of Objects from a Pile," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

[27] A. W. Fitzgibbon and R.B.Fisher, "A Buyers Guide to Conic Fitting," in *British Machine Vision Conference*, 1995, pp. 513–522.

An in-field magnetometer calibration method for IMUs

Sarvenaz Salehi, Navid Mostofi, and Gabriele Bleser

***Abstract*— Magnetometers are one of the most common aiding sensors used in Inertial Measurement Units (IMU) to provide motion tracking, the latter being an important task in Cognitive Assistive (CA) systems. The basis for accurate motion tracking is a well calibrated sensor. Therefore, this paper presents an easy-to-use method for in-field magnetometer calibration and alignment with inertial sensors in an IMU. In the first step of the calibration procedure bias, scale factors and non-orthogonality parameters are estimated based on magnitude information and data collected under motion. In the second step misalignment parameters are obtained from inclination using gravity measured by accelerometers under static conditions. In each step the initial estimation of parameters based on linear least squares followed by a nonlinear optimization leads to reliable results for all the calibration parameters. Different performance aspects of the method are evaluated in several tests using real data.**

I. INTRODUCTION

Motion tracking plays a significant role in several tasks of CA systems, such as activity monitoring and user feedback. Tracking of body and hand postures provides important pieces of activity information [1],[2]. Head motion tracking enables Augmented Reality feedback visualized through Head Mounted Displays (HMDs) [3],[4],[5]. Using body-worn IMUs in these applications removes the need for external sensing infrastructure and can reduce latency and computational costs.

Given a valid calibration, IMUs comprising triads of gyroscopes, accelerometers and magnetometers can provide accurate orientation in three dimensions. While gyroscopes measure angular velocities, accelerometers, under moderate body accelerations, provide a vertical reference and the earth magnetic field vector measured by magnetometers is a good reference in the horizontal plane. In contrast to the inertial sensors, magnetometer measurements are easily disturbed by ferromagnetic materials i.e. hard/soft iron [6] in the vicinity of the sensor, either inside the IMU casing or in the final mounting position. This results in errors in the magnetometer measurements after installation at the customer's site, e.g. when mounting the device on an HMD. As discussed in [7] these errors can be interpreted as changes of the calibration parameters such as biases, scale factors, non-orthogonality of the axes, and misalignment of the magnetometer coordinate frame with the inertial one, which are usually pre-calibrated by the manufacturer. This can significantly deteriorate the tracking performance. In order to compensate for such errors, an accurate in-field calibration of the magnetometers is required which motivates the present work. It should be mentioned that this paper covers the time-invariant disturbances that move with the sensor frame while external effects caused by hard/soft iron materials external to the IMU coordinate frame need to be handled online, e.g. either by explicitly estimating these disturbances or by rejecting respective measurements [8],[9].

The authors are with the department Augmented Vision, German Research Center for Artificial Intelligence, Trippstadter Str. 122, 67663 Kaiserslautern, Germany
{Sarvenaz.Salehi, Navid.Mostofi, Gabriele.Bleser}@dfki.de.

The paper is organized as follows: Section II describes related work. Section III summarizes the proposed method, which is then detailed in Section IV (calibration parameters) and Section V (algorithm implementation), and evaluated in Section VI. Conclusions are drawn in Section VII.

II. RELATED WORK

Different methods for magnetometer calibration have been proposed in literature. Using precise external references such as Helmholtz coils for the magnetometer calibration [10] is expensive and cannot be practical for all types of applications. Other proposed methods use the earth magnetic field, sometimes in combination with attitude information. Swinging is a traditional attitude dependent method which requires a number of known headings in the horizontal plane [11]. A simpler approach is to use only information concerning the earth magnetic field such as field magnitude (scalar checking) or inclination. Using field magnitude an ellipsoid fitting approach is proposed in [12] which is, however, based on minimizing algebraic distances and leads to a suboptimal estimation of the calibration parameters. Vasconcelos et al. in [7] propose a geometric approach to obtain an optimal estimate of the calibration parameters using a maximum likelihood estimator (MLE). However, the method turns out to be sensitive to how it is initialized, where the latter is not uncommon when working with real data. There is also a solution for the alignment problem proposed, which however, requires external information sources. Alonso and Shuster in [13] present a complete calibration procedure based on scalar checking and a so-called centering approach. Although misalignment is proven to be unobservable from the magnitude, the proposed work lacks a solution for obtaining these parameters. Besides the field intensity, another bit of information concerning the earth magnetic field is its inclination which can be obtained from gravity measured by accelerometers under static conditions. Hu et al. in [14] use an ellipsoid fitting method for bias estimation, and a solution based on inclination obtained from accelerometer measurements is provided to estimate the remaining calibration parameters in one coefficient matrix. However, the accuracy of this method in terms of scale factors depends on the location on earth. At the equator, where magnetic field vector and gravity are orthogonal, the scale factors are unobservable and their estimation degrades when approaching the equator. Moreover, additional noise and errors caused by the usage of accelerometer data captured under motion result in a degraded estimate of the respective parameters. The Software Development Kit (SDK) of the well-established commercial IMUs from Xsens [21]

provides a magnetic field mapping procedure, which can, however, not be used with other IMUs. In our work we provide a complete solution to the considered problem together with a comparative performance evaluation with respect to some of the previously discussed approaches.

III. Approach

The present work proposes an easy-to-use procedure and an algorithm for in-field magnetometer calibration and alignment to the coordinate system of an IMU. The procedure is attitude independent and works without any need for precise equipment or external heading information. It is based on the assumption that the magnetic field is homogeneous during the calibration, i.e. the measured earth magnetic field vectors have constant magnitude and inclination independently of the IMU pose. In order to properly extract this information from the IMU measurements, the manual procedure is divided into two steps with different data capturing approaches. In the first step, as suggested in [12], bias, scale factors and non-orthogonality parameters are estimated using ellipsoid fitting and a set of magnetometer measurements recorded under motion. After having established an orthogonal coordinate system in this step, a rotation which aligns this system with the IMU coordinate system given by the inertial sensors is achieved in the second step. This step is based on the assumption of constant inclination and uses a set of accelerometer measurements under different static poses, thus removing errors due to body acceleration. In contrast to the method in [14], the proposed approach for the scale factor estimation is independent of the location on earth, since the magnetometer calibration parameters are purely estimated from the magnetometer measurements in the first step. The presented method provides reliable parameter estimation which is confirmed by several experiments comparing the results with two calibration methods provided with commercial IMUs.

IV. Parameters

A. Magnetometer calibration parameters

As shown in [12], in order to convert the sensor readings, $\vec{H}_m$, to the true magnetic field, $\vec{H}_t$, both in the magnetometer coordinate system, each axis should be corrected for bias, scale factor, and non-orthogonality according to the following model:

$$\vec{H}_t = CS^{-1}(\vec{H}_m - \vec{b}), \quad (1)$$

where $\vec{b}$ defines the bias vector, S is a diagonal matrix containing the scale factors, and C is a lower triangular matrix used for non-orthogonality correction as suggested in [7].

B. Misalignment parameters

Equation (1) establishes an orthogonal magnetometer coordinate system. In order to relate the calibrated magnetometer measurements, $\vec{H}_t$, to the inertial measurements, namely acceleration and angular velocities, the respective coordinate systems need to be aligned. Since the IMU frame and the magnetometer frame are both orthogonal, the alignment consists in a rotation, which can be parameterized in a minimal way by using an axis-angle representation [15]. The resulting model for converting $\vec{H}_t$ to an aligned vector, $\vec{H}_{at}$, is then given by:

$$\vec{H}_{at} = Rot(\vec{\omega})\vec{H}_t, \quad (2)$$

where $\vec{\omega}$ contains the axis-angle parameters of the rotation and $Rot(\vec{\omega})$ denotes its conversion to a rotation matrix.

V. Calibration Method

As mentioned before, the calibration method consists of two steps. In the first step the IMU is sampled while it is manually rotated in different directions to obtain a sufficient coverage of the ellipsoid. The magnetometer calibration parameters are then calculated from the captured raw magnetometer measurements. In the second step the IMU is sampled while it is static and posed in different directions. The misalignment parameters are then estimated using the captured magnetometer measurements (calibrated with the results of the first step) and the acceleration measurements (providing gravity under static conditions).

A. First step

Assuming a homogeneous field such as the earth magnetic field without disturbances, the locus of the true magnetometer measurements in the sensor frame is on the surface of a sphere with the center at the origin and the radius equal to the intensity of the local magnetic field. This sphere is deformed to an ellipsoid as the effect of biases, scale factors and non-orthogonality [12]. Therefore compensation of these effects can be posed as a problem of fitting an ellipsoid to the raw measurements by minimizing the sum of squared geometric distances [16], for which a nonlinear least squares optimization technique is required. Using equation (1), the minimization problem is defined as:

$$\text{argmin}_{C,S,\vec{b}} \sum_{i=1}^{n} \left(\left\| CS^{-1}\left(\vec{H}_{m,i} - \vec{b}\right) \right\| - H \right)^2, \quad (3)$$

where n is the number of magnetometer measurements. The constant H equals the local earth magnetic field intensity, which can be chosen according to the given location. Here, it is set to unit length, since only the direction of the magnetic vector is important for motion tracking application.

Typically, a non-convex optimization technique requires a good initial guess. This is achieved by the linear least squares approximation proposed in [12] which is based on Singular Value Decomposition (SVD). Since it has been experienced that the initial guess is easily perturbed due to sensor noise under realistic calibration conditions, different

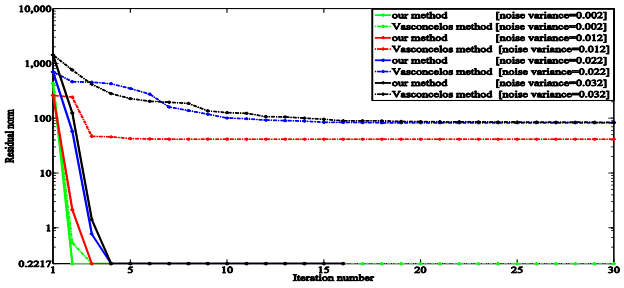


Fig. 1: This figure illustrates the development of the residuals in relation to the performed iterations when starting the nonlinear estimation from the same initial guess. When adding higher noise levels to the initial guess, the method of Vasconcelos tends to diverge.

optimization tools have been investigated in a pre-study concerning their sensitivity to such disturbances. Our final method uses the Levenberg Marquart Algorithm (LMA) [17], which turned out to perform considerably more robust than the Newton method as used by Vasconcelos et al. in [7](cf. Fig. 1 for selected results of the pre-study).

B. Second step

In order to obtain the misalignment parameters, we used inclination which is defined as the angle between the earth magnetic field vector and the earth horizontal plane and varies in different geographical locations [18]. However, this angle can be considered constant for a local area, where the calibration procedure takes place. Using the fact that the gravity vector, $\vec{g}$, is always orthogonal to the earth horizontal plane, the following equation is derived:

$$\vec{g}^T \vec{H}_{at} = \cos(\frac{\pi}{2} - \alpha), \quad (4)$$

where α denotes the local inclination angle. Here $\vec{g}$, as obtained from the accelerometers under static conditions, and $\vec{H}_{at}$ are assumed to be normalized and given in the IMU coordinate frame.

Substituting (2) in (4) gives a nonlinear optimization problem, which is again solved using the LMA:

$$\text{argmin}_{\vec{\omega},\alpha} \sum_{i=1}^{n} \left(\vec{g}_i^T Rot(\vec{\omega}) \vec{H}_{t,i} - \cos(\frac{\pi}{2} - \alpha) \right)^2, \quad (5)$$

where n is the number of measurements, $\vec{H}_{ti}$ is the normalized i[th] calibrated magnetometer measurement, and $\vec{g}_i$ is the normalized i[th] calibrated gravity measurement.

In order to obtain an initial guess, the nonlinear constraints in (4) can be relaxed using $R := Rot(\vec{\omega})$ and $c = \cos(\frac{\pi}{2} - \alpha)$. This results in an equation system, which is linear with respect to R and c and can be solved for these parameters using SVD:

$$\vec{g}_i^T R \vec{H}_{t,i} - c = 0 \qquad (i = 1, \dots, n). \quad (6)$$

Parameter c provides an initial guess for the inclination, while initial values for $\vec{\omega}$ are obtained by orthogonalizing R and then converting it to the axis-angle representation.

VI. Experimental Results

A. Test setup

The proposed calibration algorithm was implemented and tested using the measurements of a commercially available IMU [19], which includes a two axes magnetometer (MS2100) combined with a single axis magnetometer (SEN-Z65) both manufactured by PNI, and a tri-axes accelerometer (ADXL345) manufactured by Analog Devices. The IMU measurements were sampled at 100 Hz through a USB transceiver.

Fig. 2 illustrates the effects and working principles of the magnetometer calibration, visualizing the uncalibrated measurements in comparison to the calibrated ones. Ideally, after calibration, the data points should lie on the surface of a unit sphere located in the origin of the IMU coordinate system, i.e. the vectors should have unit length independently of the IMU pose.

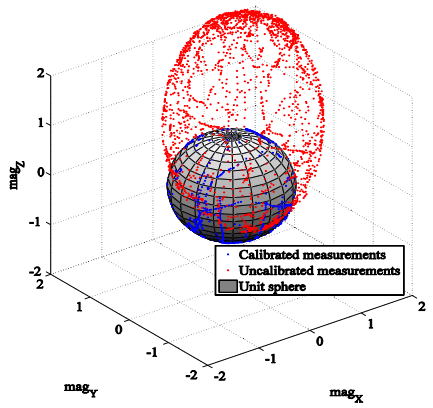


Fig. 2:Magnetometer measurements before (red) and after (blue) calibration by our proposed method. In order to simplify comparison the uncalibrated measurements are scaled to the average magnitude. Ideally, the calibrated measurements should map to a unit sphere.

To quantify the performance of the proposed calibration procedure in terms of accuracy, different criterions were defined and evaluated. First, deviation from unit length (magnitude deviation) was evaluated based on calibrated data sequences captured under arbitrary IMU rotations. Then, the direction accuracy of the calibrated magnetometer measurements was investigated in terms of heading error and plane projections based on known motions. These were performed in a special test setup as illustrated in Fig. 3 and consisted in rotations around the three axes of a high-precision aluminum cube, the latter being aligned with the IMU mounted inside.

Using the above criterions, the results of the proposed algorithm were compared with the results of the calibration method provided with the SDK of the IMU, which corresponds to the aforementioned method of Hu et al [14], and with the calibrated magnetometer measurements of the Xsens MTi, which is a well established commercial IMU.

Before capturing the evaluation sequences, both IMUs were calibrated in an outdoor area (in Kaiserslautern, Germany) in order to assure a homogeneous magnetic field. The calibration of the Xsens MTi was performed based on the instructions in [20]. Calibration parameters for the Trivisio IMU were obtained from both, the method provided in the SDK (subsequently called Hu method) and the proposed method. For both calibrations almost the complete ellipsoid surface, the locus of magnetometer measurements, was covered by 3000 samples, which proved to be a suitable number of samples to obtain reasonable results. An evaluation of how the coverage of measurements affects the ellipsoid fitting is presented in [7].

Since the proposed calibration algorithm is based on a manual procedure for data capturing, in addition to the accuracy evaluation above, the repeatability of the method was assessed in terms of variation of the bias parameters when performing the calibration task several times.

Fig. 3: A high precision aluminum cube on a turntable made of glass and aluminum with a flat wooden plate as a leveled base served as test setup for performing known motions. Here, the Xsens MTi is mounted inside the cube.

B. Results

Before discussing the results of the different accuracy evaluations as described before, it is worth demonstrating the advantage of the proposed algorithm over Hu et al., based on a dataset captured in a location close to the equator. As mentioned before, when being based on inclination information, the scale factor estimation degrades significantly as the inclination angle approaches zero degrees. The effect is clearly visible in Fig. 4, where the measurements calibrated with Hu deviate significantly from the unit sphere, while the measurements calibrated with the proposed method are well aligned. This result proves the independence of the proposed scale factor estimation method from the location on earth. The reason is that in our method biases, scale factors and non-orthogonality parameters are purely obtained from magnetometer measurements based on magnitude information, whereas inclination is only used for the calibration of misalignment. Despite this clear superiority over Hu, it has to be noted that one degree of freedom of the misalignment matrix is still affected by a horizontal or vertical magnetic field vector, given at the equator or the poles, respectively.

1) Magnitude deviation from unit length

Fig. 5 shows the magnitude results on a data sequence of 5000 samples, while the IMUs were rotated around various axes. The average magnitude of our method is 0.99, of Xsens is 0.98 and of Hu et al. is 1.11. Hence, our method provides

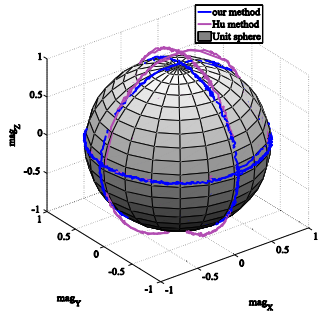


Fig. 4: Magnetometer measurements calibrated with our method (blue) and the method of Hu et al (purple). The dataset was sampled in a location close to the equator.

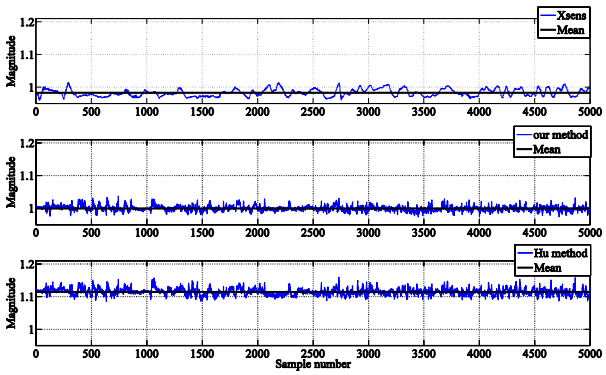


Fig. 5: Magnitudes of the magnetic field vectors measured under a wide range of rotations and calibrated with the three methods.

comparable results to Xsens and outperforms Hu. This test only evaluates the first step of the calibration procedure, purely related to the magnetometers. As a side note, the plots indicate that the magnetometer measurements obtained from the Xsens IMU are prefiltered, which is not the case for the Trivisio IMU.

The experiment can also be used to compute the expected level of magnitude variation, which can be helpful in distinguishing valid magnetometer measurements from outliers in a real-time orientation tracking scenario.

2) Angular error

As mentioned before, angular errors were determined using the test setup of Fig. 3. After mounting the respective IMU in the aluminum cube, heading changes of 90 degrees were performed in three different attitudes. Given the known poses, the average magnetometer vector measured in the first pose was considered as reference and rotated accordingly and then compared to the measured vectors in each pose. The test conditions for both IMUs were the same, and for each static pose 100 measurements were sampled. Since in most orientation tracking applications it is important to track the heading change, this test evaluates the change of heading without requiring local heading values as reference. Fig. 6 shows the angular errors between the measured and the rotated vectors in each pose.

Our method has an average error of 1.6° (max=3.54°) which is lower in comparison with 2.07° (max=3.69) for Xsens and 1.92° (max=4.67) for Hu. This experiment evaluates both sets of parameters, the magnetometer related

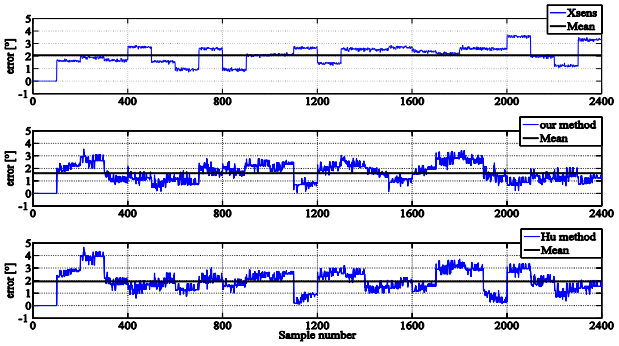


Fig. 6: Angular errors resulting from the three calibration methods.

parameters and the alignment with the inertial coordinate frame. The results show a good performance of our method, which is slightly better than the other methods, and proves that the magnetometer measurements are well calibrated and aligned with the IMU. It has to be noted that the results obtained from the Xsens are not strictly comparable due to the mounting of the two individual IMUs.

3) Plane projections

As a last accuracy test, the two IMUs were rotated around the x-, y-, and z-axes and the projections of the calibrated measurements on the three coordinate planes yz, xz, and xy were investigated. The results are illustrated in Fig. 7.

Ideally, the projected measurements should form a circle. In order to quantify the similarity to a circle, we computed the eccentricity of the ellipse fitted to each set of projected data, which should be close to zero. For the three data sets the eccentricities, [around x, around y, around z], for our method equals to [0.07,0.07,0.07], for Hu equals to [0.16,0.11,0.16], and for Xsens is [0.17,0.08,0.14]. These results again demonstrate a good performance of our method. This experiment primarily evaluates the quality of the non-orthogonality and misalignment parameter estimates. The eccentricity obviously increases if these parameters are not considered in the calibration procedure. Since the rotations have been performed on the planes of the aluminum cube, which are assumed to be aligned with the IMU coordinate frame, the resulting projections are distorted, if the axes of the magnetometers are not well aligned. As in the previous experiment, the same note concerning the Xsens IMU should be considered.

Another insight from this experiment concerns the radii of the ellipses resulting from the inclination but also the estimated scale factors. While the results obtained from our method and the Xsens are comparable, the Hu method seems to overestimate the scale factors. This effect was already apparent in the magnitude experiment and confirms a degraded performance of the scale factor estimation being based on acceleration measurements.

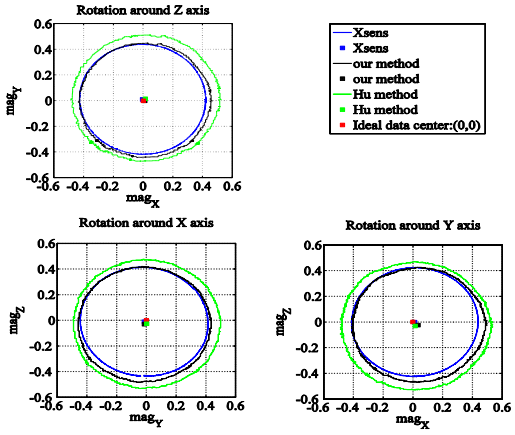


Fig. 7: Projections of the calibrated magnetometer measurements on the different coordinate planes using the three calibration methods.

4) Repeatability

In order to provide an indication for the repeatability of our method, we collected ten datasets of 3000 samples each while varying the manual rotations of the IMU maintaining a sufficient coverage. We then calculated the calibration parameters from each of the collected datasets and investigated their variation. The standard deviation of the biases (which are easiest to interpret) for the three axes in relation to the range of measurements in these tests was calculated with [0.07, 0.036, 0.35] percent. Even though the numbers are difficult to interpret, they indicate that variations in the manual data capturing process result in repeatable calibration parameters.

VII. Conclusion

This paper presents and evaluates an in-field magnetometer calibration method for IMUs requiring neither precise external equipment nor local magnetic field information. The proposed method is complete, as it obtains biases, scale factors, non-orthogonality and misalignment parameters. Moreover, it is easy to perform, since the user only has to rotate the IMU around various axes for about 30 seconds and to place it in a few different static poses.

During the implementation process, special attention was brought to choosing an optimization tool, which is insensitive to a perturbed initial guess. The latter was found to be a real problem in practical application [7].

The obtained calibration parameters were evaluated in different test scenarios demonstrating in most cases higher precision in comparison with the calibration method included in the commercial SDKs of the two available IMUs. In particular it could be shown, that our proposed method performed well in all experiments demonstrating a consistent calibration of all the parameter sets. It should be mentioned that recently an algorithm for magnetometer calibration was presented by Xsens in [22] which we intend to compare to the present work in future.

Additionally the repeatability of the method was investigated in terms of the variation in the estimated bias parameters when performing the calibration tasks multiple times. The results indicate that in a homogeneous magnetic field, different ways of data capturing have only minor influence on the calibration parameters.

The performed evaluations also provide indications for online filtering and outlier rejection of the magnetometer measurements in a motion tracking scenario which is the subject of our future work.

Acknowledgements

This work has been partly performed within the project COGNITO (ICT - 248290) funded under the 7th framework program (ICT-2009.2.1). Moreover, the authors would like to thank J.F. Vasconcelos and F. Ludwig for providing an implementation of their algorithm [7] for investigating the convergence properties.

References

[1] M. Weber, G. Bleser, G. Hendeby, A. Reiss, and D. Stricker, "Unsupervised Model Generation for Motion Monitoring", IEEE

SMC Workshop on Robust Machine Learning Techniques for Human Activity Recognition, Anchorage, AK, USA, 2011.

[2] S. Sundaram and W. Cuevas, "High level activity recognition using low resolution wearable vision", In Workshop on Egocentric Vision, pages 25–32, 2009.

[3] G. Schall, D. Wagner, G. Reitmayr, E. Taichmann, M. Wieser, D. Schmalstieg, and B. Hofmann-Wellenhof, "Global pose estimation using multi-sensor fusion for outdoor augmented reality", In Proc. ISMAR'09, 2009.

[4] E. Foxlin, "Head tracking relative to a moving vehicle or simulator platform using differential inertial sensors", In Proc. SPIE AeroSense Conf. Helmet- Head-Mounted Displays V, pages 133–144, Orlando, 2000.

[5] S. You, U. Neumann, and R. Azuma, "Hybrid Inertial and Vision Tracking for Augmented Reality Registration", In Proc. of IEEE VR '99, pages 260-267, 1999.

[6] E. R. Bachmann, X. Yun, and A. Brumfield, "Investigating the effects of magnetic variations on inertial/magnetic orientation sensors", IEEE ICRA '04, Volume 2, pages 1115–1122, 2004.

[7] J. Vasconcelos, G. Elkaim, C. Silvestre, P. Oliveira and B. Cardeira, "A geometric approach to strapdown magnetometer calibration in sensor frame", In Proc. IFAC Workshop on Navigation, Guidance, and Control of Underwater Vehicles (NGCUV), Killaloe, Ireland, 2008.

[8] A. M. Sabatini, "Quaternion-based extended Kalman filter for determining orientation by inertial and magnetic sensing", IEEE Trans. Biomed. Eng.,vol. 53, no. 7, pages 1346–1356, 2006.

[9] D. Roetenberg, H. J. Luinge, T. M. Baten, and P. H. Veltink, "Compensation of magnetic disturbances improves inertial and magnetic sensing of human body segment orientation", IEEE Trans. Neural Syst. Rehab.Eng., vol. 13, no. 3, pages 395–405, 2005.

[10] J. C. Springmann, J. W. Cutler, and H. Bahcivan, "Magnetic sensor calibration and residual dipole characterization for application to nanosatellites", In Proc. AIAA/AAS Astrodynamics Specialist Conference, Toronto, 2010.

[11] N. Bowditch, "The American Practical Navigator.Defense Mapping Agnecy, Hydrographic/TopographicCenter", Bethesda, MD, 1995.

[12] V. Renaudin, M. H. Afzal, and G. Lachapelle, "New method for magnetometers based orientation estimation", In IEEEIION PLANS Coriference, 2010.

[13] R. Alonso and M. D. Shuster, "Complete linear attitude-independent magnetometer calibration", The Journal of the Astronautical Sciences, vol. 50, no. 4, pages 477–490, 2002.

[14] X. Hu, Y. Liu, Y. Wang, Y. Hu, and D. Yan. "Autocalibration of an electronic compass for augmented reality" In Proc. ISMAR 2005, pages 182-183, Vienna, Austria, 2005.

[15] M. D. Shuster, "A survey of attitude representations" The Journal of the Astronautical Sci., vol. 41, pages 439–517, 1993.

[16] W. Gander, G. H. Golub, and R. Strebel, "Least-Square Fitting of Circles and Ellipses", BIT, no. 43, pages 558-578, 1994.

[17] J. More, "The Levenberg-Marquardt Algorithm, Implementation, and Theory", Numerical Analysis, G. A. Watson, ed., Springer-Verlag, 1977.

[18] National Geophysical Data Center (NGDC), http://www.ngdc.noaa.gov/ngdc.html (accessed at 23/08/2012)

[19] http://www.trivisio.com/index.php/products/motiontracking/colibriwireless (accessed at 23/08/2012)

[20] Magnetic Field Mapper MT Manager Add-on User Manual, © 2008, Xsens Technologies B.V.

[21] http://www.xsens.com/ (accessed at 23/08/2012)

[22] M Kok, J Hol, T Schn, F Gustafsson, and H Luinge, "Calibration of a magnetometer in combination with inertial sensors", International Conference on Information Fusion, 2012.