Project no.:	**027657**
Project full title:	**Perception, Action & Cognition through Learning of Object-Action Complexes**
Project Acronym:	**PACO-PLUS**
Deliverable no.:	**D7.1.3**
Title of the deliverable:	**Action Selection for Robotic Manipulation of Deformable Objects**

Contractual Date of Delivery to the CEC:	**31 January 2009**	
Actual Date of Delivery to the CEC:	**24 February 2009**	
Organisation name of lead contractor for this deliverable:	**CSIC**	
Author(s): Juan Andrade-Cetto, Saúl Cuén and Carme Torras		
Participants(s): CSIC		
Work package contributing to the deliverable:	**WP7**	
Nature:	**R/D**	
Version:	**1.0**	
Total number of pages:	**18**	
Start date of project:	**1st Feb. 2006**	**Duration: 48 month**

Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)
Dissemination Level

PU	Public	**X**
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (including the Commission Services)	

Abstract:

This deliverable contains one scientific publication describing a technique to manipulate fabric objects for kitchen oriented applications. In particular, a piece of cloth is straightened by selecting robot actions that both get the job done and minimize state uncertainty at the same time. This method for choosing actions is designed to act over *i*OACs at the middle layer of the PACO-PLUS cognitive architecture.

Keyword list: manipulation planning, model-based planning.

Table of Contents

INTRODUCTION

This deliverable deals with the manipulation of planar deformable objects such as fabric, for typical service robot applications. The action selection mechanisms described in this deliverable have been designed to reduce the uncertainty in the estimation of the attributes in an instantiated OAC; in this case, the *i*OAC is the state space representation of the piece of cloth. This is achieved by selecting from a set of primitive actions at the middle level of the PACO-PLUS cognitive architecture; the ones that maximize the predicted mutual information gain between posterior states and measurements. Maximizing the mutual information helps to avoid ill-conditioned measurements.

The essential idea is to use mutual information as a measure of the statistical dependence between actions and attributes in the *i*OAC. The mutual information is the relative entropy between the marginal density of the attributes and the same density conditioned on the observed attribute values. When the attributes are modeled as multivariate Gaussian distributions, the parameters of the marginal density are trivially a Kalman filter prior mean and covariance. Moreover, the parameters of the conditional density come precisely from the Kalman update equations.

As suggested in the Y2 Review Report, this entropy-based action selection mechanism is now integrated at the middle level of the PACO-PLUS cognitive architecture, as detailed in Deliverable 4.2.3 and publications therein. The issue of computational complexity, raised also in the Review Report, although not reported here has been dealt with by using an information form representation of Gaussian distributions. The main idea is to recover efficiently joint marginals of the state pdf distributions. These marginals can then be used to evaluate the mutual information metric that quantifies the expected outcome of actions in terms of reduction of uncertainty. This can be achieved by encoding covariances and cross correlations in a tree-like structure, each node of the tree containing upper and lower bounds on these matrices computed using interval arithmetics. The method has been submitted for a related problem, but in a different context, and the reference is only included here for completeness [C]. Computational resources are in general, no longer an issue when evaluating the entropy-based action selection mechanism.

The review raised concerns also on the relevance of using deformable objects instead of the more classic rigid objects for which the outcome of actions are being characterized in PACO+. Typically, during object manipulation of rigid objects, expected outcomes can be characterized within a discrete set, related to object affordances, i.e., empty - filled, upside - uspide down, graspable, etc. The level of uncertainty that can be encoded here is too simplistic for real robot applications. Dealing with uncertainty in manipulation calls inevitably for richer object state representations, one that is only possible with continuous state representations. Flexible objects offer this richness of representation. Instead of dealing with the continuous variables of rigid object pose, we can now talk about continuous state representations for the object form, with folds and wrinkles providing a rich set of uncertainties for the action selection mechanism to deal with.

ACTION SELECTION FOR ROBOTIC MANIPULATION OF CLOTH

Service robots, and in particular those aimed at helping humans in daily tasks, are required to operate in a wide range of tasks and environments. The variability of tasks and environments in which they are to operate pose new research problems not tackled within industrial robotics. The non-repetitive manipulation of deformable objects is one such problem, since these objects are plentiful in homes and assistive environments.

While a lot of work has been devoted to grasping, motion planning and manipulation of rigid objects, similar research for deformable objects is just starting. PACO-PLUS is addressing the grasping and manipulation of both types of objects within a kitchen environment. The long-term goal is to plan and execute manipulation tasks with the fingered hands of the ARMAR robot, but as a first step this paper deals with action selection for cloth straightening with a Staubli arm.

Global action plans on what to do with an object, the piece of cloth in this case, are computed in the upper layer of the PACO-PLUS cognitive architecture. Once a plan has been decided, to straighten the piece of cloth in this case, it is passed to the middle layer for actuation over the instantiated OAC, together with a set of primitive actions, which had been learnt useful to achieve the goal through learning. At this point, local decisions must be made to minimize contingencies and to maximize reward. One such method for local plan execution is to take the actions that are most informative, in the sense that they help reduce the uncertainty in the estimation of attributes in the *i*OAC. The work reported here is in that sense related to WP5 for global planning, to WP6 for the learning of motion primitives, and to WP7 for local decision making.

Specifically, we present a system that straightens pieces of cloth from any arbitrary initial wrinkle condition using a robotic manipulator. The cloth is modeled with a finite element method, and its state is estimated with a physical-based implicit integration scheme that computes particle velocities as a function of internal and external forces acting on the object. The state of the object instantiation is tracked with a stochastic observer, in which measurements come from a stereovision system. Manipulation actions are chosen maximizing an a-optimal information measure.

In the work reported in this deliverable we have adopted an implicit integration scheme to estimate the state of a planar deformable object, given its better accuracy in estimating deformations over extended periods of time. In this approach, a large sparse linear system is solved through a preconditioned conjugate gradient (CG) iterative method. The preconditions of the CG method permit imposing external constraints on the velocities of some particles, which comes handy when we need to restrain the motion of the particles fixed by the finger.

To our knowledge, this is the first time that a stochastic state estimator has been derived for an implicit integration model of a deformable planar object, bridging the gap between computer simulation and vision-based tracking of the state of deformable planar objects for manipulation. A video showing the simulation and actual implementation of the approach is available at http://www-iri.upc.es/people/cetto/pacoplus/Manipulation_Deformable_Bodies.wmv

ATTACHED PAPER

[A] S. Cuén, J. Andrade-Cetto, and C. Torras. “Action selection for robotic manipulation of cloth”. To be published. A smaller version of this paper was presented as
[B] S. Cuén, J. Andrade-Cetto, and C. Torras. “Action selection for robotic manipulation of deformable objects”. In ESF-JSPS Conference on Experimental Cognitive Robotics, Kanagawa, March 2008.

OTHER REFERENCES

[C] V. Ila, Josep M. Porta, and J. Andrade-Cetto, “A Tree-based Nearest Neighbor Search for Data Association in Pose SLAM”, submitted to Robotics Science and Systems Conference, 2009.

Action Selection for Robotic Manipulation of Cloth

Saúl Cuén-Rochín, Juan Andrade-Cetto and Carme Torras

Institut de Robòtica i Informàtica Industrial, CSIC-UPC
Llorens Artigas 4-6, Barcelona 08028, Spain
`scuen,cetto,torras@iri.upc.edu`.

Abstract

This paper deals with the manipulation of planar deformable objects such as fabric, for typical service robot applications. Specifically, we present a system that straightens pieces of cloth from any arbitrary initial wrinkle condition using a robotic manipulator. The cloth is modeled with a finite element method, and its state is estimated with a physical-based implicit integration scheme that computes particle velocities as a function of internal and external forces acting on the object. The state of the object is tracked with a stochastic observer, in which measurements come from a stereo vision system. Manipulation actions are chosen maximizing an a-optimal information measure.

To our knowledge, this is the first time that a stochastic state estimator has been derived for an implicit integration model of a deformable planar object, bridging the gap between computer simulation and vision-based tracking of the state of deformable planar objects for manipulation.

1 Introduction

Service robots, and in particular those aimed at helping humans in daily tasks, are required to operate in a wide range of tasks and environments. The variability of tasks and environments in which they are to operate pose new research problems not tackled within industrial robotics. The non-repetitive manipulation of deformable objects is one such problem, since these objects are plentiful in homes and assistive environments.

While a lot of work has been devoted to grasping, motion planning and manipulation of rigid objects [16, 18], similar research for deformable objects is just starting [1, 31]. The European Project PACO-PLUS [3] is addressing the grasping and manipulation of both types of objects within a kitchen environment. The long-term goal is to plan and execute manipulation tasks with the fingered hands of the ARMAR robot, but as a first step this paper deals with action selection for cloth straightening with just one finger.

The existing work on manipulation of deformable objects deals mainly with linear objects [9, 25, 17], such as ropes threads and wires. These works usually rely on a Finite Element Method (FEM) to model the objects, and make use of knot topology to plan motions. Modeling deformable planar objects –those of interest to us– in the same way may be computationally costly, and the alternative of using a Boundary Element Method (BEM) has been proposed [12], where BEM differs from FEM in that only the contour of the object needs to be meshed. However, BEM-based simulation does not provide enough detail on cloth state for our purposes, so we do not adopt this method in the present work. Other works focus on grasping skills for cloth manipulation [26], and on iterative learning of the force required to lift a deformable object [14]. A compilation of systems for the industrial manipulation of deformable objects is discussed in [13], going from sewing systems to fish manipulation processes.

For action planning, a physical-based simulation that accurately predicts the outcome of actions is crucially needed. In the Computer Graphics field, there are two approaches to cloth simulation that use FEM to describe the particles positions and velocities as a mesh of primitives such as triangles and rectangles. One is the implicit integration scheme [5], which at the expense of a high computational cost, remains stable despite taking long time steps. The other is the explicit integration scheme [23], with lower computational burden, but restrained to take short time steps to assure stable solutions. The reader is referred to [19] for a detailed discussion of cloth simulation approaches, and to [11] for a wider compilation of physical models of deformable objects.

In the current work we have adopted the implicit integration scheme, given its better accuracy in estimating the state of the object over extended periods of time. In this approach, a large sparse linear system is solved through a preconditioned conjugate gradient (CG) iterative method. The preconditions of the CG method permit imposing external constraints on the velocities of some particles, which comes handy when we need to restrain the motion of the particles fixed by the finger.

State estimation techniques are used to track the state of the cloth. In particular, an Extended Kalman filter on the implicit integration scheme has been implemented. The selection of the best next action to straighten the cloth is then tackled using tools from information theory. This approach to action selection has previously been pursued in the context of active vision for Simultaneous Localization and Mapping [29], and wire-based robot pose tracking [2]. Here, we have adapted it to cope with the uncertainty inherent to the manipulation of deformable objects.

The paper is structured as follows. Section II details the model used to predict the deformation of cloth under the presence of three types of forces: i) internal forces such as stretch, shear and bend, ii) the forces exerted by our manipulation strategy, and iii) external forces such as gravity and the collision with other objects. In Section III the action selection strategy is described. The strategy has the dual objective of straightening the cloth while at the same time maintaining good estimation of its state. Section IV presents both simulated and experimental results, and Section V contains some concluding remarks.

2 Physical Model

To model its deformation, a piece of cloth may be modeled as a triangular mesh of particles. We use a uniform particle distribution on a square plane shape with disk topology for our cloth representation, compatible to the one typically used in the computer graphics community [5].

Each vertex of the triangular mesh has coordinates $\mathbf{p}_i$, and moves with velocity $\mathbf{v}_i$, The state of the cloth be defined as a $6n$-dimensional array containing the vertical concatenation of all vertex locations $\mathbf{p}$, and velocities $\mathbf{v}$, respectively.

To model the deformation of the entire mesh after a time step h, we adopt a variable-velocity motion model given by the difference equation

$$\begin{bmatrix} \mathbf{p}_{t+h} \\ \mathbf{v}_{t+h} \end{bmatrix} = \begin{bmatrix} \mathbf{p}_t + h\mathbf{v}_{t+h} \\ \mathbf{v}_t + \Delta\mathbf{v} \end{bmatrix} \tag{1}$$

The change in velocity for the interconnected particles, $\Delta\mathbf{v}$, follows the backward Euler method for implicit time integration given in [5]. Baraff's paper explains in depth how to form and solve the equation. The method, in contrast to a more simple forward Euler integration technique, finds an output state whose time derivative is consistent with the initial state. The method is used to simulate the effect of internal and external forces applied to the cloth. These forces acting on each particle are defined in terms of precondition functions that permit to impose constraints on the velocities, effectively allowing us to model the effect of motion commands on those particles fixed by the manipulator gripper:

$$\Delta\mathbf{v} = \mathbf{A}^{-1}\mathbf{b} \tag{2}$$

$$\mathbf{A} = \mathbf{I} - h\mathbf{W}\frac{\partial \mathbf{f}}{\partial \mathbf{v}_t} - h^2\mathbf{W}\frac{\partial \mathbf{f}}{\partial \mathbf{p}_t}$$

$$\mathbf{b} = h\mathbf{W}\left(\mathbf{f}_t + h\frac{\partial \mathbf{f}}{\partial \mathbf{p}_t}\mathbf{v}_t\right) + \mathbf{u} + \delta\mathbf{u}$$

The solution for $\Delta\mathbf{v}$ depends on the initial particle velocities $\mathbf{v}_t$, the $3n \times 3n$ particle constrainer matrix $\mathbf{W}$, whose block diagonal elements are defined as $\mathbf{W}_{ii} = \frac{1}{m_i}\mathbf{S}_i$, for which m_i is the mass of the i-th particle, and $\mathbf{S}_i$ is a 3×3 matrix used to constraint the desired degree of freedom affecting the particle mass at any given location. When particles are constrained, $\mathbf{u}$ is used to set their desired velocity, and $\delta\mathbf{u}$ are external induced error velocities which will come handy in robotic manipulation. Forces acting on the system include the initial particle vector forces $\mathbf{f}_t$, that may include external forces such as gravity, wind, etc., and the internal forces $\mathbf{f}$, which according to the material characteristics can be expressed as functions of the particle positions $\mathbf{p}_t$. Particle internal forces are modeled as the sum of resistance and damping effects on specific stretch (C_u, C_v) , shear (C_s) and bend (C_b) conditions

$$\mathbf{f} = \sum_{i=1}^{4}\left[-\mathbf{k}_i\frac{\partial \mathbf{C}_i(\mathbf{p}_t)}{\partial \mathbf{p}_t}\mathbf{C}_i(\mathbf{p}_t) - \mathbf{d}_i\frac{\partial \mathbf{C}_i(\mathbf{p}_t)}{\partial \mathbf{p}_t}\dot{\mathbf{C}}_i(\mathbf{p}_t)\right]$$

where

$$\dot{\mathbf{C}}_i(\mathbf{p}_t) = \frac{\partial \mathbf{C}_i(\mathbf{p}_t)}{\partial \mathbf{p}_t}\mathbf{v}_t$$

$$\mathbf{C}(\mathbf{p}_t) = (C_u, C_v, C_s, C_b)^\top$$

and $\mathbf{k} = (k_u, k_v, k_s, k_b)^\top$ and $\mathbf{d} = (d_u, d_v, d_s, d_b)^\top$ are appropriate resistance and damping factors, respectively.

2.1 Internal Forces

In Baraff's formulation [5], $\mathbf{C}(\mathbf{p})$ is a condition vector which we want to be zero. Its associated energy $E = \frac{k}{2}\mathbf{C}(\mathbf{p})^\top\mathbf{C}(\mathbf{p})$ is used to derive simple stretch, shear and bend conditions. So, for example if $\mathbf{w}_u(\mathbf{p})$ and $\mathbf{w}_v(\mathbf{p})$ are the vectors indicating the stretch or compression of the triangles in the mesh, with unit length when the material is straightened, and a is the area of the triangle in uv fixed planar coordinates, the condition

$$\begin{bmatrix} C_u \\ C_v \end{bmatrix} = a\begin{bmatrix} \|\mathbf{w}_u(\mathbf{p})\| - 1 \\ \|\mathbf{w}_v(\mathbf{p})\| - 1 \end{bmatrix}$$

can be used to model stretch energy. Similarly, by the small angle approximation, shear can be measured as the inner product between $\mathbf{w}_u(\mathbf{p})$ and $\mathbf{w}_v(\mathbf{p})$

$$C_s = a\mathbf{w}_u(\mathbf{p})^\top \mathbf{w}_v(\mathbf{p}) \, .$$

Finally, if we let $\mathbf{n}_i$ and $\mathbf{n}_j$ denote the unit normals of two adjacent triangles, and let $\mathbf{e}$ be a common vector parallel to the common edge, the angle between the two faces defined by the relations $\sin\theta = (\mathbf{n}_i \times \mathbf{n}_j)^\top \mathbf{e}$ and $\cos\theta = \mathbf{n}_i^\top \mathbf{n}_j$, the condition that counters bending along that edge is

$$C_b = \theta \, .$$

2.2 Influence of the Parameters

In [6] and [20] studies are performed to determine the stretch and damping parameters $\mathbf{k}$ and $\mathbf{d}$ for the physical model presented in this paper. We use the same parameters as in [20] to adequate physical animation of a soft fabric. Nonetheless, a change in parameters may be done to adequate animation to different materials such a flexible thin sheet of metal.

2.3 Motion commands

Our cloth dynamics model, the time varying partial differential Equation (2), differs from the original equation in [5] in that we have included a set of external induced error velocities $\delta\mathbf{u}$. We use $\mathbf{u}$ to represent the actions exerted by our manipulator, and for the effects of linearization and unmodeled artifacts on each external action on the cloth we use the stochastic term $\delta\mathbf{u}$ with zero mean white Gaussian distribution with covariance $\mathbf{Q}$.

Input commands belong to a limited set of actions depending on the task to be performed. Table 1 shows, for example, a set of possible actions for the straightening of a cloth on the table. Each such action is intended to drag a corner in the cloth at a constant speed and for a short period of time, $\mathbf{u}_i = (v_x, v_y, 0)^\top$.

2.4 Managing Collisions

The two mainstreams for cloth or deformable planar object collision response are the preventive [7] [8] and corrective [4] [30] [24] methods. The cloth collision response community agrees that the use of preventive methods complemented with corrective ones when the former fails is recommendable. We use

Table 1
Set of Possible Actions to straighten a piece of cloth.

Action	
`drag-upright`	$v_x, v_y > 0$
`drag-upleft`	$v_x < 0, v_y > 0$
`drag-downright`	$v_x > 0, v_y < 0$
`drag-downleft`	$v_x, v_y < 0$

a preventive technique [7] in our implementation, which takes a good friction aproximation. The method simulates solid external objets with large mass, making the solid mass collision points much bigger than the cloth triangles, modifying only the cloth particle positions and velocities when collisions are detected. To avoid $O(n^2)$ comparisons, we use a bounding box approach. The reader is referred to [28] for a thorough discussion on collision detection.

Collision detection and collision response is modeled as an extra term, independent of the integration method of choice. It is expressed as a function

$$\Delta \mathbf{v}_c(\mathbf{p}_t, \mathbf{v}_t, \mathbf{p}_{t+h}, \mathbf{v}_{t+h})$$

which delivers the change in velocity needed to avoid internal or external collisions with respect to collision free positions $\mathbf{p}_t$, collision free velocities $\mathbf{v}_t$, step foward position $\mathbf{p}_{t+h}$, and velocities $\mathbf{v}_{t+h}$, that had not been taken into account during collision response. This gives us an enriched model to estimate the deformation of cloth over time that includes cloth/cloth and solid/cloth collision response:

$$\begin{bmatrix} \mathbf{p}_{c,t+h} \\ \mathbf{v}_{c,t+h} \end{bmatrix} = \begin{bmatrix} \mathbf{p}_t + h\mathbf{v}_{c,t+h} \\ \mathbf{v}_t + \Delta\mathbf{v} + \Delta\mathbf{v}_c \end{bmatrix} \tag{3}$$

where $\mathbf{p}_{c,t+h}$ and $\mathbf{v}_{c,t+h}$ are the collision free positions and velocities in the next time step, respectively.

3 Action Selection

3.1 Predicting the Outcome of Actions

To estimate the state of the deformable planar object after an action is executed, particle positions and velocities are considered as a random vector with Gaussian distribution and initial covariance $\mathbf{P}_{0|0}$. A Kalman filter is then used to track its state. For every possible action, the state mean can be obtained from Eqs. (1), (2) and (3) with $\delta \mathbf{u} = \mathbf{0}$, and an estimated change in covariance can be computed with the linearized expression

$$\mathbf{P}_{t+h|t} = \mathbf{F}\mathbf{P}_{t|t}\mathbf{F}^\top + \mathbf{G}\mathbf{Q}\mathbf{G}^\top .$$

To this end, the plant Jacobians with respect to the state

$$\mathbf{F} = \begin{bmatrix} \frac{\partial \mathbf{p_{t+h}}}{\partial \mathbf{p_t}} & \frac{\partial \mathbf{p_{t+h}}}{\partial \mathbf{v_t}} \\ \frac{\partial \mathbf{v_{t+h}}}{\partial \mathbf{p_t}} & \frac{\partial \mathbf{v_{t+h}}}{\partial \mathbf{v_t}} \end{bmatrix}$$

and the plant/noise Jacobian

$$\mathbf{G} = \begin{bmatrix} \frac{\partial \mathbf{p_{t+h}}}{\partial \delta \mathbf{u}} \\ \frac{\partial \mathbf{v_{t+h}}}{\partial \delta \mathbf{u}} \end{bmatrix} = \begin{bmatrix} h\mathbf{I} \\ \mathbf{I} \end{bmatrix}$$

need be evaluated.

During state prediction for action selection, collision response is not computed. The computational burden of the evaluation of $\Delta \mathbf{v}_c$ for every possible action is unfeasible. Instead "incomplete" Jacobians are evaluated, and small model discrepancies are dealt with as noise. Predictions are revised with sensor reading to compensate such gap.

The state of the object can then be revised from the observation of some points as measured by our stereo vision system. Assuming that the error from our sensor $\delta \mathbf{z}_i$ is also zero mean Gaussian with covariance $\mathbf{R}$, each particle observed, whereas it is a corner or not, contributes in revising the state estimate with

$$\begin{bmatrix} \mathbf{p}_{t+h|t+h} \\ \mathbf{v}_{t+h|t+h} \end{bmatrix} = \begin{bmatrix} \mathbf{p}_{t+h|t} \\ \mathbf{v}_{t+h|t} \end{bmatrix} + \mathbf{K}(\mathbf{z}_t - \mathbf{p}_t) .$$

If using sequential innovation for each particle, the measurement Jacobian is a row block of zeros, only with a selective 3×3 identity matrix at the i-th block cell, and the Kalman gain becomes the $6n \times 3$ matrix

$$\mathbf{K} = \mathbf{P}_{t+h|t,i}(\mathbf{P}_{t+h|t,ii} + \mathbf{R})^{-1}$$

where $\mathbf{P}_{t+h|t,ii}$ is the position covariance for the i-th particle, and $\mathbf{P}_{t+h|t,i}$ represents the i-th column block of the full state covariance matrix. The state covariance update becomes

$$\mathbf{P}_{t+h|t+h} = (\mathbf{I} - [\,\mathbf{0}_{6n\times 3i-1}\ \mathbf{K}\ \mathbf{0}_{6n\times 3(2n-i)-2}\,])\mathbf{P}_{t+h|t} .$$

Table 2
Computational Cost For Position Tracking.

Grid Size	Vertices	Computation Time
6 x 6	36	1.71s
7 x 7	49	3.40s
8 x 8	64	6.64s
9 x 9	81	11.85s
10 x 10	100	25.09s
11 x 11	121	41.66s

3.2 Action Selection

A strategy is developed to straighten the cloth by choosing from a limited set of possible actions, the one that maximizes the information gain for our state estimate. The set of actions under inspection are one possible motion command from Table I for each corner of the object. The commands evaluated are those that drive the cloth corners away from the center. In essence, the strategy is aimed at choosing, from four possible choices, which corner is to be dragged next, based on the current estimate that we have about its location.

A classic approach would be to chose the action that maximizes the relative entropy between prior and posterior covariance estimates [2, 15, 29], that for our multivariate Gaussian case reduces to computing the expression

$$I = \frac{1}{2}(\log |\mathbf{P}_{t+h|t}| - \log |\mathbf{P}_{t+h|t+h}|)\,.$$

This D-optimality measure of information gain however may become unreliable when one or more of the state space directions is constrained, since it is computed from the product of the eigenvalues of $\mathbf{P}$. The conditions that constraint the motion of particles include contact with an obstacle, or the mere dragging action. In such cases, there is absolute information about some components of the location (and/or velocity) of such particle, with the consequence of semi-definteness on the estimation covariance. For this reason, we use an A-optimality measure of information instead [27] to minimize the squared error of the model, which computes the sum of the eigenvalues instead of their product

$$I = \text{tr}\ (\mathbf{P}_{t+h|t}) - \text{tr}\ (\mathbf{P}_{t+h|t+h}) \tag{4}$$

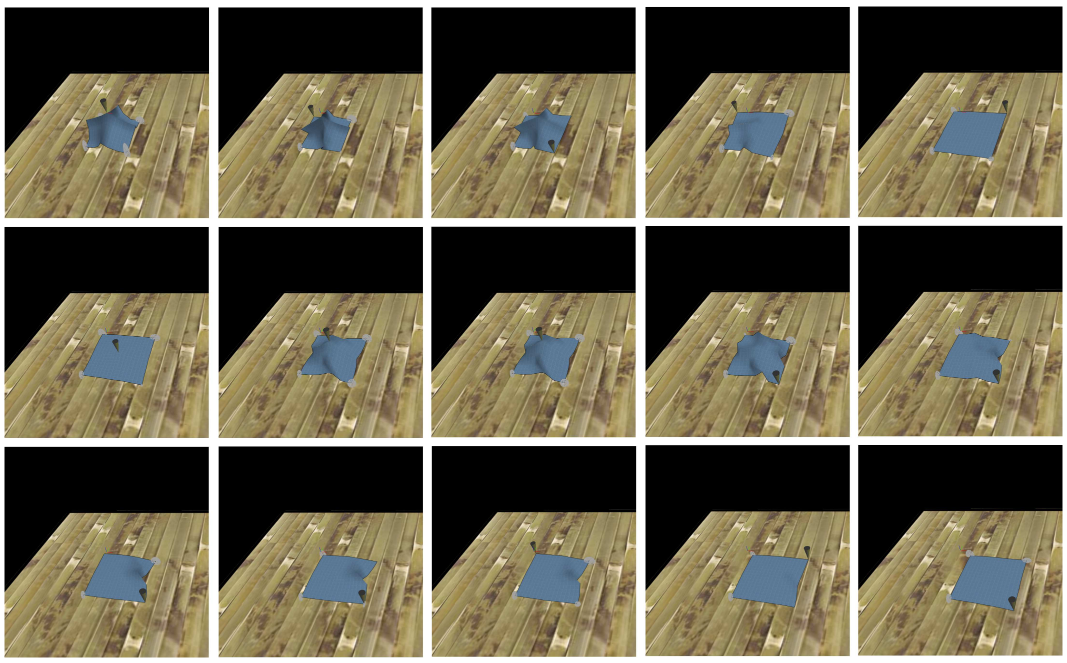

Fig. 1. Computer simulation of action selection of planar deformable objects. Time goes from left to right, then from top to bottom. The hyper-ellipsoids on the corners indicate surfaces of equal probability for the corner location estimates.

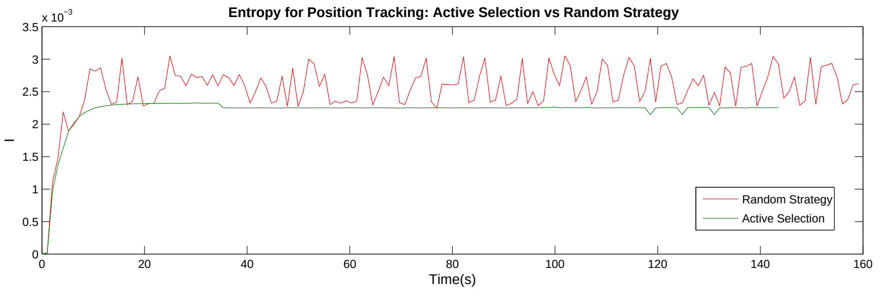

Fig. 2. Active Selection vs Random Selection

Table 3
Parameter values for implementation.

Parameter	Symbol	Value
Stretch resistance	k_u, k_v	5000
Shear resistance	k_{sr}	500
Bend resistance	k_b	0.00001
Stretch damping	d_u, d_v	1000
Shear damping	d_{sr}	100
Bend damping	d_b	2×10^{-6}
Gravity		9.81

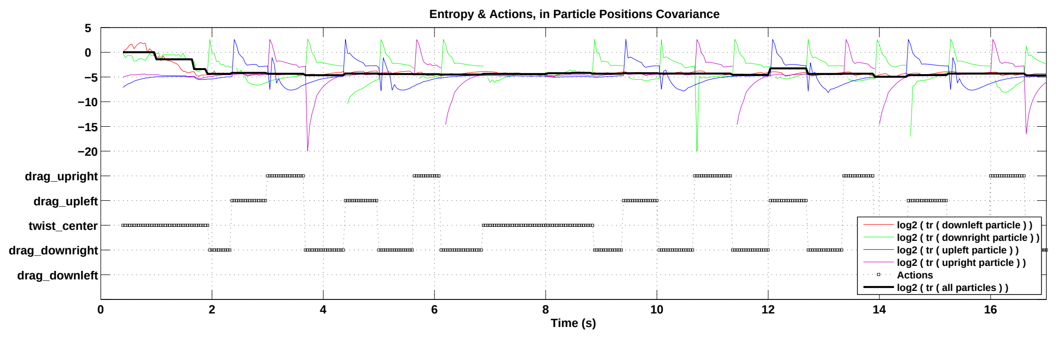

Fig. 3. Evolution of the trace of the covariance at each particle for the same simulation, as well as a history of the chosen motion commands

4 Implementation and Results

In our experiment setup, our workcell is composed of a robotic manipulator *Stäubli RX-60* with a *FTC-Schunk* force sensor attached to the end-effector, and a *Bumblebee* stereo camera. The force sensor is used to ensure that a sufficiently large perpendicular force is maintained while dragging a piece of cloth against a table.

To measure the state of the cloth at any given instance, a set of feature points must be observed. One possibility is to select scale invariant salient features on the object and match them against a previously trained data-set [22]. The experiments reported here are less complex in terms of the computer vision tools used. For the purpose of our straightening task, we are content with tracking the four corners of the cloth piece binarizing the image of the object, and detecting the corners by selecting the discontinuities of the 1-D signal along the object contour using multiresolution and non-maximum suppression [10]. Once the corner points are located, their location with respect to the camera is computed with stereo triangulation. Given that the camera location is calibrated with respect to the robot workcell, the measurement of the corner points can be given in world coordinates.

To constraint the velocity of the particles during manipulation, a velocity profile must be generated for each possible action command. The set of possible actions is restricted to the actions in Table I. For the simulations however, an extra motion `twist` has been implemented, which drags two center particles to emulate a wrinkling effect to restart the simulation.

The cloth parameters used are given in Table 3. Fig. 1 shows a simulated manipulation sequence. Each frame in the sequence represents the current state of the deformable object. The ellipsoids drawn at the corners represent surfaces of equal probability at one-standard deviation, and are used to indicate the value of the estimation covariance at that point. To easy their visibility,

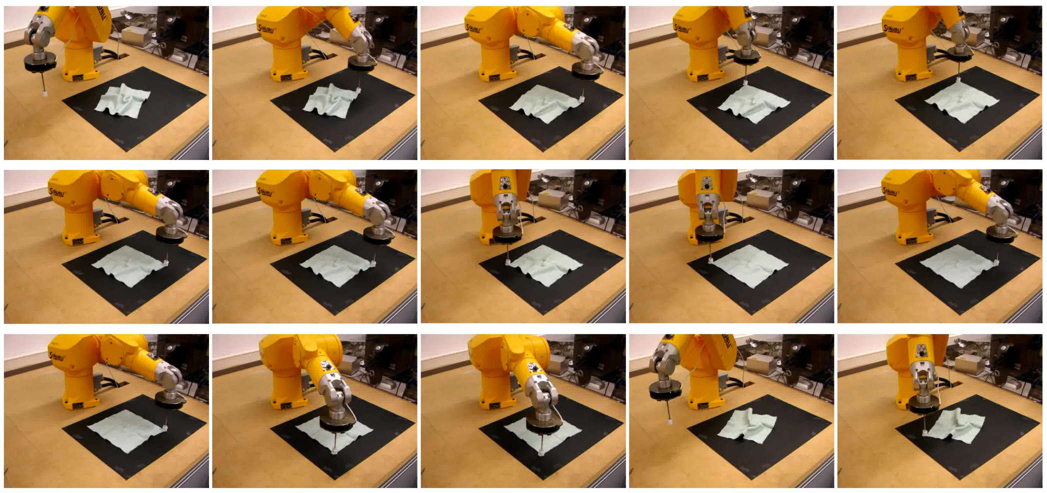

Fig. 4. A view of the workcell setup. Time goes from left to right, then from top to bottom

these ellipsoids have been magnified by a factor of ten. The sequence shows two instances of the evolution of the cloth straightening task. The cone pointing to each of the corners emulates the manipulator end-effector. Figure 3 contains the evolution of the trace of the covariance at each particle for the same simulation, as well as a history of the chosen motion commands.

Figure 2 shows the comparison beet-wen the active selection strategy and a random action selection strategy. The experiment shows how the uncertainty reduction strategy keeps the entropy I stable for the duration of the experiment.

Figure 4 presents a sequence with a real straightening task on our robotic workcell. These images, as well as the video accompanying this paper illustrate the feasibility of the presented approach for the information-oriented action selection for the manipulation of planar deformable objects for simple household applications.

The tracker was implemented in C++ and the physical model implementation based on the free software [21]. The system runs on a Mobile Intel Pentium 4, 1.8GHz, 512MB DDR SDRAM. The physical model runs smooth at 30fps with a 11 x 11 Grids (121 vertices), and the computational time to track the position and take the manipulation active action selection is shown in Fig. 2 not including the time spent on collision detection.

5 Conclusions and Future Work

If the adaptive robot manipulation of rigid objects is already a challenging research topic, the manipulation of deformable objects poses additional difficulties. An important one is the representation of the state of such objects. For rigid objects, once a CAD model of the object is available, its state at a given time is uniquely determined by the six parameters of its pose. Contrarily, flexible objects require models that accommodate their possible deformations as a result of their manipulation or other external causes. In this work we have coupled a stochastic state estimator with a physical-based implicit integration model of a deformable planar object (a cloth). This model has then been used to predict the effect of manipulation actions on the cloth, a critical feature for planning sequences of such actions. Here, as a first step to test state estimation, only the best next action to achieve a given goal is determined. The particular goal pursued has been a weighted combination of two objectives, namely, straightening the cloth while at the same time maintaining a good estimation of its state. Action selection relies on a maximization information criterion. The obtained results, both in simulation and in a real robotic work-cell, have been satisfactory. In sum, we are proposing a framework for goal-driven manipulation of deformable planar objects.

Envisaged future work will be along five lines. First, we aim to come up with a characterization of qualitative-different states of deformable planar objects (e.g., foldings), in a similar way as knots describe states of deformable linear objects. Second, we like to go beyond single action selection to develop planning strategies for manipulating pieces of cloth, more specifically, for unfolding and then folding them in prescribed ways. Third, multirobot action selection manipulation. Forth, learning the object material parameters with neural network approaches such as in [14] with the help of vision system approaches such as in [22].

6 Acknowledgements

This work is supported by the EU PACO-PLUS project FP6-2004-IST-4-27657 and by the Spanish Ministry of Science and Innovation project DPI2008/06022.

References

[1] K. Hamajima abd M. Kakikura. Planning strategy for task of unfolding clothes. *Robot. Auton. Syst.*, 32(2-3):145–152, Aug. 2000.

[2] J. Andrade-Cetto and F. Thomas. A wire-based active tracker. *IEEE Trans. Robot.*, 24(3):642–651, Jun. 2008.

[3] J. Andrade-Cetto and C. Torras. PACO-PLUS: Perception, action and cognition through learning of object-action complexes. In *Proc. 2nd. Jornada de Recerca en Automática, Visió i Robòtica*, pages 265–272, Barcelona, Jul. 2006.

[4] D. Baraff, A. Witkin, and M. Kass. Untangling cloth. *ACM Trans. Graphics*, 22:862–870, 2003.

[5] D. Baraff and A.P. Witkin. Large steps in cloth simulation. In *Computer Graphics. Proc. ACM SIGGRAPH Conf.*, pages 43–54, Orlando, Jul. 1998. ACM Press.

[6] K. Bhat, C. Twigg, J. Hodgins, P. Khosla, Z. Popovic, and S. Seitz. Estimating cloth simulation parameters from video. In *Proc. ACM SIGGRAPH/EUROGRAPHICS Sym. Comput. Anim.*, pages 37–51, San Diego, 2003.

[7] R. Bridson, R. Fedkiw, and J. Anderson. Robust treatment of collisions, contact and friction for cloth animation. *ACM Trans. Graphics*, 21(3):594–603.

[8] R. Bridson, S. Marino, and R. Fedkiw. Simulation of clothing with folds and wrinkles. In *Proc. ACM SIGGRAPH/EUROGRAPHICS Sym. Comput. Anim.*, pages 28–36, San Diego, 2003.

[9] J. Brown, J. C. Latombe, and K. Montgomery. Real-time knot-tying simulation. *Visual Comput.*, 20(2-3), 2004.

[10] R. Deriche. Fast algorithms for low-level vision. *IEEE Trans. Pattern Anal. Machine Intell.*, 12(1):78–88, Jan. 1990.

[11] S. Gibson and B. Mirtich. A survey of deformable modeling in computer graphics. Technical Report TR-97-19, MERL, Cambridge, Nov. 1997.

[12] M.A. Greminger, Y. Sun, and B.J. Nelson. Boundary element deformable object tracking with equilibrium constraints. In *Proc. IEEE Int. Conf. Robot. Automat.*, pages 3896–3901, New Orleans, Apr. 2004.

[13] D. Henrich and H. Wörn, editors. *Robot Manipulation of Deformable Objects.* Advanced Manufacturing. Springer, 2000.

[14] A. M. Howard and G. A. Bekey. Intelligent learning for deformable object manipulation. *Auton. Robot.*, 9(1):51–58, 2000.

[15] D. J. C. MacKay. Information based objective functions for active data selection. *Neural Comput.*, 4(4):589–603, 1992.

[16] M. T. Mason. *Mechanics of Robotic Manipulation.* MIT Press, 2001.

[17] M. Moll and L. E. Kavraki. Path planning for deformable linear objects. *IEEE Trans. Robot.*, 22(4):625–636, 2006.

[18] R. M. Murray, Z. Li, and S. Shankar Sastry. *A Mathematical Introduction to Robotic Manipulation.* CRC Press, 1994.

[19] H. Ng and R. Grimsdale. Computer graphics techniques for modeling cloth. *IEEE Comput. Graphics Applicat.*, 16(5):28–41, Sep. 1996.

[20] D. Pritchard. Cloth parameters and motion capture. Master's thesis, University of British Columbia, Vancouver, Canada, 2003.

[21] D. Pritchard. Freecloth 0.7.1. a free, open-source cloth simulation tool, 2003.

[22] D. Pritchard and W. Heidrich. Cloth motion capture. In *Proc. Eurographics*, volume 22, pages 263–271, Granada, Sep. 2003. Blackwell Publishing.

[23] X. Provot. Deformation constraints in a mass-spring model to describe rigid cloth behavior. In *Proc. Graphics Interface*, pages 147–154, Quebec, 1995. Canadian Human-Computer Communications Society.

[24] X. Provot. Collision and self-collision handling in cloth model dedicated to design garments. In *Proc. Eurographics Workshop Comput. Anim. Simul.*, pages 177–190, Budapest, Sep. 1997.

[25] M. Saha and P. Isto. Motion planning for robotic manipulation of deformable linear objects. In *Proc. IEEE Int. Conf. Robot. Automat.*, pages 2478–2484, Orlando, May 2006.

[26] K. Salleh, H. Seki, Y. Kamiya, and M. Hikizu. Tracing manipulation of deformable objects using robot grippers with roller fingertips. In *Proc. SICE-ICASE Joint Conf.*, pages 5882–5887, Busan, Oct. 2006.

[27] R. Sim and N. Roy. Global A-optimal robot exploration in SLAM. In *Proc. IEEE Int. Conf. Robot. Automat.*, pages 673–678, Barcelona, Apr. 2005.

[28] M. Teschner, S. Kimmerle, B. Heidelberger, G. Zachmann, L. Raghupathi, A. Fuhrmann, M.-P. Cani, F. Faure, N. Magnenat-Thalmann, W. Strasser, and P. Volino. Collision detection for deformable objects. *Computer Graphics Forum*, 24(1):61–81, Mar. 2005.

[29] T. Vidal-Calleja, A.J. Davison, J. Andrade-Cetto, and D.W. Murray. Active control for single camera SLAM. In *Proc. IEEE Int. Conf. Robot. Automat.*, pages 1930–1936, Orlando, May 2006.

[30] M. Wicke, H. Lanker, and M. Gross. Untangling cloth with boundaries. In *Proc. 11th Int. Fall Workshop Vision, Modell., Visualization*, pages 349–356, Aachen, Nov. 2006.

[31] G. T. Zoumponos and N.A. Aspragathos. Fuzzy logic path planning for the robotic placement of fabrics on a work table. *Robot. Comput.-Integr. Manuf.*, 24(2):174–186, Apr. 2008.

A Tree-Based Nearest Neighbor Search for Data Association in Pose SLAM

Viorela Ila, Josep M. Porta and Juan Andrade-Cetto. Paper-ID 82

Abstract— **The most expensive operation in the information form of Pose SLAM, the variant of SLAM where only the robot trajectory is estimated, is state recovery after loop closure. However, by rigorously controlling the number of loop closures we obtain a system where the robot operates most of the time in open loop, amortizing the cost of loop closure over long trajectories. In this case, the bottleneck for information-based Pose SLAM shifts to data association and, in particular, to the search for previously visited poses close to the current one that are good candidates for sensor registration. During the estimation of the distance between candidate matching poses cross correlations need to be computed. As the robot moves, these cross correlations change and state-of-the-art approaches to recover them without approximations scale at least linearly with the number of states. Such bound on complexity limited until now the need for a neighboring pose search strategy that performs better than a basic linear scan over all previous poses. In this paper, we show that the cross covariances between poses can be factorized in terms of the previous and current poses, and that such factorization can be updated in constant time as the robot moves. Exploiting this property, it makes sense to speed up the nearest neighbor search. We achieve such speed up by organizing state means and variances in a balanced tree and defining internal levels of such tree using interval arithmetic. The result is a strategy to search for neighboring poses that scales logarithmically with the number of states. We present results both on simulated and real data that validate the approach.**

I. Introduction

Solutions to the Simultaneous Localization and Mapping (SLAM) problem have evolved over the years, both with respect to the type of state entries that are maintained, as well as with respect to the estimation form used. Seminal SLAM solutions relied on Kalman filtering to estimate the absolute position of landmarks and the robot pose, at the expense of quadratic computational cost [1], limiting its use only to map small areas. If instead, a canonical representation of Gaussians is used, the resulting information matrix turns out to be approximately sparse, i.e., the matrix entries for distant landmarks are very small and the matrix can be sparsified with a minimal information loss, trading optimality for efficiency [2].

Efficiency without information loss is possible with exact sparse representations. Using the information form, this can be achieved estimating the entire robot path along with the map, an approach typically referred to as full SLAM [3, 4, 5]. Exact sparsification is also possible if only one set of variables is maintained; either by keeping a small set of active landmarks kidnapping and relocating the robot [6], by decoupling the estimation problem maintaining the map only [7], or as it is done in Pose SLAM, by maintaining only the pose history using landmarks solely to produce relative measurements between robot poses [8, 9].

Pose SLAM can be approached as a graph optimization problem for which solutions are found using batch processes that iteratively search for the maximum log likelihood of the entire graph represented as quadratic constraints [10, 11, 12], or by factorizing the sparse information matrix [4]. Incremental approaches to Pose SLAM are based either on variants of the previous batch methods [5] or on filtering [8, 13]. In this case, only constant time is needed for predictions and updates for linear (or linearized) systems.

It is known, however, that the linearizations introduced at each iteration in both the extended Kalman and information filters forms of SLAM produce overconfident estimates, which in the long run lead to filter inconsistency [14, 15]. One way to defer filter inconsistency in Pose SLAM is to control the number and type of loops to be closed [13], closing only few highly informative loops. Similar approaches have been used in landmark-based SLAM where inconsistency is delayed incorporating only highly informative observations to the filter [16, 17]. Moreover, if only a small number of loop closures is maintained, the space and time complexity of Pose SLAM scales better. Applying this strategy, the robot closes only few loops and, between loop closures, it operates in open loop for long periods, which is feasible using recent odometric techniques [18, 19].

When closing a loop, the state can be recovered with time complexity close to $n\,log(n)$, with n the number of poses [8, 20]. This computational cost is amortized over the period where the robot operates in open loop, for which the filter is updated in constant time. In this context, the bottleneck for real time execution is not state recovery but data association, that is, detecting neighboring poses for which feature matching is likely. Data association is typically implemented as a linear search, either by directly searching for feature matches in a sensor database, independent of the filter estimates [21], or by first using filter information to constrain the search for sensory matches [8, 13]. For consistent filter estimates, the second option is more efficient and less prone to perceptual aliasing. However, the estimation of the distance between two poses requires the computation of their joint marginal, something that is not directly available in information-based filtering schemes.

Efficient approximations of cross covariances can be computed in logarithmic time by subsampling poses and performing relaxation over multiple spatial resolutions [11], or in constant time by considering only first order relations via Markov blankets [2] or by implementing partial state updates [22]. Optimistic approximations of joint marginals increase the number of data association candidates, something that is especially sensitive after long periods of open loop traverse. Thus, exact cross covariance computation is preferred for the accurate identification of nearest neighbors. These can be recovered by augmenting the sparse system of equations needed for state recovery [8] or by exploiting the sparseness of factorized forms of the information matrix with QR [5] or Cholesky factorizations [23]. These algorithms have in average linear computational complexity for band diagonal matrices, but can be more expensive for matrices encoding many loops. With these time complexity bounds, it did not make sense to implement a search for neighboring poses better than the basic linear scan over all previous poses.

In this paper we show that, when operating in open loop, exact joint marginals can be computed in constant time. It makes sense then to have more efficient methods to search for data association. In the work reported here this is achieved using a tree structure that finds nearest neighbors in $O(\log n)$. While tree-like structures have been used to represent the solution space for data association in landmark-based SLAM [24], to our knowledge, no existing approach uses a tree to represent the search space for data association in Pose SLAM. This opens the possibility to use Pose SLAM for large scale problems that involve long trajectories.

The paper is structured as follows. In Section II we succinctly formalize Pose SLAM including the data association problem. Then, in Section III, a strategy to compute marginal joint covariances in constant time during open loop is described. Building on this result, Section IV describes a method to determine data association hypotheses in logarithmic time. Section V shows the computational advantage of the technique when compared to a linear scheme for data association by means of simulating very large trajectories as well as on real data sets. Concluding remarks are given in Section VI.

II. Identification of Neighboring Poses

We consider the recursive form of Pose SLAM, where the objective is to estimate the trajectory of the robot at time t, $\mathbf{x}_t = \{x_0, \ldots, x_t\}$, with x_i the robot pose at time i. Using a typical Bayesian recursion, $\mathbf{x}_t$ is updated with a set, $\mathbf{z}_t$, of observations of the relative displacement between the current robot pose and previous poses along the trajectory

$$p(\mathbf{x}_t|\mathbf{z}_t, \mathbf{x}_{t-1}) \propto p(\mathbf{x}_t|\mathbf{x}_{t-1})\, p(\mathbf{z}_t|\mathbf{x}_t). \tag{1}$$

The observations at time t can be split in two disjoint groups, a set of observations between the current robot pose and the immediate previous one, $\mathbf{u}_t$, and a set of observations linking the current pose with any other pose but the previous one, $\mathbf{y}_t$. With this we have

$$\begin{aligned} p(\mathbf{x}_t|\mathbf{z}_t, \mathbf{x}_{t-1}) &\propto p(\mathbf{x}_t|\mathbf{x}_{t-1})\, p(\mathbf{u}_t, \mathbf{y}_t|\mathbf{x}_t) & (2)\\ &\propto p(\mathbf{x}_t|\mathbf{x}_{t-1})\, p(\mathbf{u}_t|\mathbf{x}_t)\, p(\mathbf{y}_t|\mathbf{x}_t) & (3)\\ &\propto p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t)\, p(\mathbf{y}_t|\mathbf{x}_t). & (4)\end{aligned}$$

The set $\mathbf{u}_t$ is the subset of observations used for state augmentation, i.e., for open loop operation. This set can be easily integrated into an information filter in constant time using the prediction phase of the filter.

Integrating the observations in $\mathbf{y}_t$ is more problematic since they usually require a data association step first that detects possible links between the current robot pose and any other previous pose along the trajectory. Data association relies on testing for the matching of sensor readings obtained at the poses to be linked. However, this is computationally demanding and the information in the filter can be exploited to restrict this test to a small number of hypotheses that can be characterized as follows.

The relative displacement, d, from the current robot pose $x_t \sim \mathcal{N}(\mu_t, \mathbf{\Sigma}_{tt})$ to any other pose $x_i \sim \mathcal{N}(\mu_i, \mathbf{\Sigma}_{ii})$ can be estimated as a Gaussian with parameters

$$\mu_d = h(\mu_t, \mu_i), \tag{5}$$

$$\mathbf{\Sigma}_d = [\mathbf{H}_t\, \mathbf{H}_i] \begin{bmatrix} \mathbf{\Sigma}_{tt} & \mathbf{\Sigma}_{it}^\top \\ \mathbf{\Sigma}_{it} & \mathbf{\Sigma}_{ii} \end{bmatrix} [\mathbf{H}_t\, \mathbf{H}_i]^\top, \tag{6}$$

where h is the measurement function giving the relative motion between the two robot poses, $\mathbf{H}_t$ and $\mathbf{H}_i$ are the Jacobians of h with respect to the two poses, and $\mathbf{\Sigma}_{it}$ is the cross correlation between poses i and t.

Marginalizing the distribution of the displacement, d, for each one of its dimensions, r, a one-dimensional Gaussian distribution $\mathcal{N}(\mu_r, \sigma_r^2)$ is obtained that allows to compute the cumulative probability of pose i being closer than v_r to pose t along this dimension

$$\begin{aligned} p_r &= \int_{-v_r}^{+v_r} \mathcal{N}(\mu_r, \sigma_r^2) \\ &= \frac{1}{2}\left(\operatorname{erf}\left(\frac{v_r - \mu_r}{\sigma_r\sqrt{2}}\right) - \operatorname{erf}\left(\frac{-v_r - \mu_r}{\sigma_r\sqrt{2}}\right)\right). \end{aligned} \tag{7}$$

If for all dimensions, p_r is above a given threshold, s, then pose x_i is considered to be close enough to the current robot pose, x_t, and it becomes a good candidate to test for pose association using the sensor readings. Before matching sensor readings, the returned poses can be ranked with respect to their statistical relevance in updating the system. That is, possible loop closures can be ranked with respect to their contribution in updating the filter, for instance, with respect to an information gain criteria [13].

III. Computation of the Exact Joint Marginals

If the similarity relation in Eq. (7) is to be evaluated for any arbitrary pair of poses we would need to have access to their joint marginals. In other words, we would need access to the whole covariance matrix which, as mentioned, is only feasible for small problems. In practice, Eq. (7) is only evaluated for the current robot pose and any previous pose from the trajectory. For this, it is enough to store the block-diagonal and the last block column of the covariance matrix [25]. However, this last block column has to be extended and updated at each prediction step, with a computational cost that is, at least, linear with the size of the history of states. Next, we present a way to obtain these marginal covariances and cross covariances in constant time when operating in open loop. That is, when data association is necessary.

Suppose a loop closure occurs at time l. At that point, and thanks to the sparsity of the information matrix, exact marginal covariances $\mathbf{\Sigma}_{ii}$ and cross covariances $\mathbf{\Sigma}_{il}$ with $1 \leq i \leq l$, can be recovered practically in near linear time either by QR or Cholesky factorization [5, 23]. In our implementation we make use of supernodal sparse Cholesky factorization [20, 26].

After the loop closure event, when the robot moves to a new pose, $x_i, i > l$, the marginal covariance for this new pose is simply a linear propagation from the previous robot pose, and can be computed as

$$\mathbf{\Sigma}_{ii} = \mathbf{F}_i\, \mathbf{\Sigma}_{i-1\, i-1}\, \mathbf{F}_i^\top + \mathbf{W}_i\, \mathbf{\Sigma}_u\, \mathbf{W}_i^\top, \tag{8}$$

with $x_i = f(x_{i-1}, u_i)$, $u_i \sim \mathcal{N}(\mu_u, \mathbf{\Sigma}_u)$ the relative displacement which brings the robot to the new pose, and $\mathbf{F}_i$ and $\mathbf{W}_i$ the Jacobians of f with respect to x_{i-1} and u_i, respectively. These covariances can be computed once and stored since they do not change until the next loop closure occurs [22].

For the cross covariances, we introduce a factorization between the last robot pose and the previously stored ones that can be updated in constant time. Algebraic manipulation renders that

$$\mathbf{\Sigma}_{it} = \mathbf{\Phi}_i\, \mathbf{F}^\top, \tag{9}$$

with

$$\mathbf{\Phi}_i = \begin{cases} \mathbf{\Sigma}_{il}, & i \leq l \\ \mathbf{\Sigma}_{ii}\, (\mathbf{F}_{l+1}^\top \ldots \mathbf{F}_i^\top)^{-1}, & i > l \end{cases} \tag{10}$$

where $\mathbf{F}$ is the accumulated Jacobian from the last loop closure to the current time slice

$$\mathbf{F}^\top = \mathbf{F}_{l+1}^\top \, \ldots \, \mathbf{F}_t^\top. \tag{11}$$

Observe that $\mathbf{F}$ can be updated in constant time as the robot moves. Moreover, all the information needed to define $\mathbf{\Phi}_i$ is available at time slice i and can be computed in constant time since the term $(\mathbf{F}_{l+1}^\top \ldots \mathbf{F}_i^\top)^{-1}$ is the inverse of the aggregated Jacobian, $\mathbf{F}^\top$, at time i.

IV. Tree-based Search for Neighboring Poses

Using the above, the similarity between robot poses in Eq. (7) can be seen as a relation between a point in the space defined at time i by μ_i, $\mathbf{\Sigma}_{ii}$, and $\mathbf{\Phi}_{ii}$, and a point in another space defined at time t by μ_t, $\mathbf{\Sigma}_{tt}$, and $\mathbf{F}$. Since these mean, covariance, and factor term computed at time i will not change during open loop traverse, we can store them in a clever way so that subsequent nearest neighbor search with different $\mathbf{F}$ factors can be sped up. The standard solution is to create a tree where nearest neighbor search can be executed in logarithmic time [27, 28, 29]. Unfortunately, there are several reasons that make standard algorithms not valid for our case. First, the similarity between two arbitrary poses is not computable since their cross covariances are not readily available and, thus, it can not be used to drive the tree construction. Second, the spaces on which the similarity function is defined are non Euclidean, invalidating the use of kd-tree based [30] or grid based [31] solutions. Finally, this similarity is not a metric relation which makes more general methods like [32] also not applicable to our problem.

The solution we propose is to organize the pose related terms in a binary tree exploiting the particular properties of the Pose SLAM problem. A leaf in the tree will store the mean, covariance and factor term associated with a particular pose (μ_i, $\mathbf{\Sigma}_{ii}$, $\mathbf{\Phi}_i$). The internal nodes of the tree have to somehow summarize the information of all leaves below them. In this way, a single test at the internal node level allows to discard large sets of poses, speeding up the search for neighbors. The usual solution in the literature is to use average prototypes [29, 33], but we can not use them since the similarity between arbitrary poses gives no information about the similarity with respect to the current pose. The option we explore to produce the internal node information is based on intervals bounding the pose information for all leaves under each node. For this option to be efficient, internal node information need to be as compact as possible and, thus, similar poses need to be grouped under the same internal tree node. Since nearby poses along the trajectory are likely to have similar marginal covariance and cross correlation with respect to the current robot pose, we organize the tree such that poses obtained in similar time slices end up in the same branch of the tree. Finally, since the tree grows on-line, it needs to be re-balanced to ensure queries are always executed in logarithmic time.

A. Bounding Pose Similarity using Interval Arithmetic

Interval arithmetic [34] is an extension of real arithmetic where operations are defined on intervals. For instance, for a couple of intervals $\overline{\underline{a}} = [\underline{a}, \overline{a}]$ and $\overline{\underline{b}} = [\underline{b}, \overline{b}]$ we have that $\overline{\underline{a}} + \overline{\underline{b}} = [\underline{a} + \underline{b}, \overline{a} + \overline{b}]$. If $a \in \overline{\underline{a}}$ and $b \in \overline{\underline{b}}$, then interval operation guarantee that $a + b \in \overline{\underline{a}} + \overline{\underline{b}}$.

We store in the internal tree nodes the hull ($\overline{\underline{\mu}}_i$, $\overline{\underline{\mathbf{\Sigma}}}_{ii}$, and $\overline{\underline{\mathbf{\Phi}}}_i$), of the means μ_i, marginal covariances $\mathbf{\Sigma}_{ii}$, and factors $\mathbf{\Phi}_i$. Using those hulls, we can compute an upper bound of the probability of the displacement to be inside the given thresholds for each dimension.

Formally, the relative displacement, d, used for data association (see Section II) can be estimated as an interval-based Gaussian with

$$\overline{\underline{\mu}}_d = h(\mu_t, \overline{\underline{\mu}}_i), \tag{12}$$

$$\overline{\underline{\mathbf{\Sigma}}}_d = [\mathbf{H}_t\, \overline{\underline{\mathbf{H}}}_i] \begin{bmatrix} \mathbf{\Sigma}_{tt} & \mathbf{F}\, \overline{\underline{\mathbf{\Phi}}}_i^\top \\ \overline{\underline{\mathbf{\Phi}}}_i\, \mathbf{F}^\top & \overline{\underline{\mathbf{\Sigma}}}_{ii} \end{bmatrix} [\mathbf{H}_t\, \overline{\underline{\mathbf{H}}}_i]^\top . \tag{13}$$

Then, an interval for the probability of the displacement in dimension r being below v_r can be readily computed as

$$\begin{aligned} \overline{\underline{p}}_r &= \int_{-v_r}^{+v_r} \mathcal{N}(\overline{\underline{\mu}}_r, \overline{\underline{\sigma}}_r^2), \\ &= \frac{1}{2} \left(\operatorname{erf}(\overline{\underline{u}}) - \operatorname{erf}(\overline{\underline{l}}) \right), \end{aligned} \tag{14}$$

```
INSERT($\mathcal{T}, i, \mu_i, \mathbf{\Sigma}_{ii}, \mathbf{\Phi}_i$)
    INPUTS:
        $\mathcal{T}$: The tree of poses.
        $i$: The label identifying the new pose.
        $\mu_i$: The mean of the pose to add.
        $\mathbf{\Sigma}_i$: The marginal covariance of the pose to add.
        $\mathbf{\Phi}_i$: The cross covariance factor term for the
            pose to add.
    OUTPUTS:
        $\mathcal{T}$: The modified tree.

 1: $n \leftarrow$ LASTNODE($\mathcal{T}$)
 2: $\mathcal{T} \leftarrow$ ADDPOSE($\mathcal{T}, n, i, \mu_i, \mathbf{\Sigma}_{ii}, \mathbf{\Phi}_i$)
 3: while not ISROOT($n$) do
 4:    $n \leftarrow$ PARENT($n$)
 5:    if BALANCEFACTOR($\mathcal{T}, n$) $> 1$ then
 6:       $\mathcal{T} \leftarrow$ LEFTROTATETREE($\mathcal{T}, n$)
 7:    end if
 8:    $\mathcal{T} \leftarrow$ UPDATENODE($\mathcal{T}, n$)
 9: end while
10: return $\mathcal{T}$
```

Algorithm 1: The insertion of a pose in the tree of poses.

with

$$\overline{\underline{u}} = \frac{v_r - \overline{\underline{\mu}}_r}{\overline{\underline{\sigma}}_r \sqrt{2}}, \tag{15}$$

$$\overline{\underline{l}} = \frac{-v_r - \overline{\underline{\mu}}_r}{\overline{\underline{\sigma}}_r \sqrt{2}}. \tag{16}$$

Since we are interested in a conservative test (i.e., a test that never discards branches of the tree that are worth exploring), we only need to compute the upper bound of $\overline{\underline{p}}_r$

$$\overline{p}_r = \frac{1}{2}\left(\mathrm{erf}(\overline{u}) - \mathrm{erf}(\underline{l})\right). \tag{17}$$

A problem of interval arithmetic is that, in many cases, operations produce an overestimation of the final result. This happens when we evaluate an expression that includes correlated subexpressions. For instance, the evaluation of $10\overline{\underline{x}} - 8\overline{\underline{x}}$ for $\overline{\underline{x}} = [1, 5]$ should result in the interval $[2, 10]$ but, using simple interval arithmetic, the result is $[-30, 42]$, since the two $\overline{\underline{x}}$ in the expression are assumed as independent variables when they are not. An excessive overestimation of the upper bounds $\overline{p}_r$ would vanish the advantage of using a tree. To minimize this risk, the expressions appearing in the data association have to be carefully re-ordered so that each interval variable appears the least possible in the final formulas. This is done in an *adhoc* basis, exploiting the structure of the $\mathbf{H}$ and $\mathbf{F}$ matrices and taking into account that only the diagonal of $\overline{\underline{\mathbf{\Sigma}}}_d$ is necessary to compute $\overline{p}_r$.

B. Building a Balanced Tree of Poses

The binary tree of poses is built initializing an empty tree, progressively adding new poses to it as the robot moves, and re-balancing the tree when necessary in a way similar to what is done with height-balanced binary search trees [35]. As

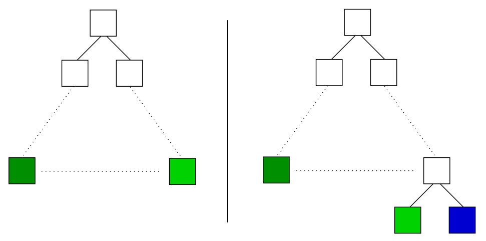

Fig. 1. Adding a new pose to the tree. White boxes represent internal tree nodes and green ones leaves. Left: The tree before the insertion. Right: The tree after adding the new pose (depicted in blue). The only pose initially in the tree affected by the insertion is the right-most one, shown in bright green in the figure.

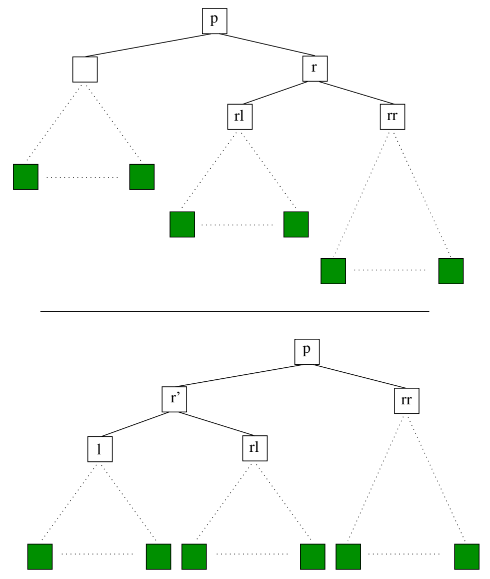

Fig. 2. Rotation of a tree to the left at node P. White boxes represent internal tree nodes and green ones leaves. Top: A tree unbalanced at node P. Bottom: The tree after applying the tree rotation.

with AVL-trees, the balance factor of a node is the difference between the height of its right and left sub-trees. A tree is considered balanced if the balance factor for all its nodes is -1. 0, or 1, where the height of a node is the maximum number of nodes from itself to the leaves.

To understand how the tree is constructed, Algorithm 1 shows the pseudo-code for the insertion operation. In this Algorithm we use the following operations

- LASTNODE($\mathcal{T}$): Returns the right-most node of the tree.
- ISROOT($\mathcal{T}, n$): Returns TRUE if n is the root node of the tree.
- ADDPOSE($\mathcal{T}, n, i, \mu_i, \mathbf{\Sigma}_{ii}, \mathbf{\Phi}_i$) Adds a new pose to the tree as depicted in Fig. 1. A new leaf with label i is created with the provided information (μ_i, $\mathbf{\Sigma}_{ii}$, and $\mathbf{\Phi}_i$) as a brother of node n generating a parent node that contains the interval hull of the information in node n and in the new node.
- PARENT(n) Returns the parent node of n.
- BALANCEFACTOR($\mathcal{T}, n$) Returns the difference in height between the right and the left sub-trees. Observe that

```
SEARCH($\mathcal{T}, \mu_t, \mathbf{\Sigma}_{tt}, \mathbf{F}, v, s$)
  INPUTS:
      $\mathcal{T}$:  The tree where to search.
      $\mu_t$:  The mean of the current robot pose.
      $\mathbf{\Sigma}_{tt}$: The marginal covariance for the current
            robot pose.
      $\mathbf{F}$:  Accumulation of Jacobians of the state
            augmentation model.
      $v$:  Maximum relative displacement from the
            current robot pose to accept a pose as a
            neighbor.
      $s$:  Minimum probability to accept a pose
            as a neighbor.
  OUTPUTS:
      $N$:  The set of labels of the neighboring poses.

 1: $i \leftarrow$ ROOT($\mathcal{T}$)
 2: $(k_i, \mu_i, \mathbf{\Sigma}_{ii}, \mathbf{\Phi}_i) \leftarrow$ GETNODEINFO($\mathcal{T}, i$)
 3: if NEIGHBOR($\mu_t, \mathbf{\Sigma}_{tt}, \mathbf{F}, \mu_i, \mathbf{\Sigma}_{ii}, \mathbf{\Phi}_i, v, s$) then
 4:     if HEIGHT($\mathcal{T}$) $= 1$ then
 5:         $N \leftarrow k_i$
 6:     else
 7:         $N_l \leftarrow$ SEARCH(LEFT($\mathcal{T}, i$), $\mu_t, \mathbf{\Sigma}_{tt}, \mathbf{F}, v, s$)
 8:         $N_r \leftarrow$ SEARCH(RIGHT($\mathcal{T}, i$), $\mu_t, \mathbf{\Sigma}_{tt}, \mathbf{F}, v, s$)
 9:         $N \leftarrow N_l \cup N_r$
10:     end if
11: else
12:     $N \leftarrow \emptyset$
13: end if
14: return $N$
```

Algorithm 2: A query using the tree of poses.

since insertions are done one at a time and always in the right-most part of the tree, the balance factor can only positive (or 0).

- LEFTROTATETREE($\mathcal{T}, n$) Increases the height of right child and decreases that of the left one as shown in Fig. 2, preserving the leaves under the node as well as their order.
- UPDATENODE($\mathcal{T}, n$) Redefines the information in an internal tree node as the interval hull of the information stored in the root node of its right and left sub-trees.

All operations during insertion have constant time complexity. However, UPDATENODE needs to be applied to the nodes from the insertion point all the way to the root. Since the tree is balanced, the total cost of inserting a new pose is $O(\log n)$ with n the number of poses already in the tree. Moreover, the memory used by the tree scales with $O(n)$. Finally, note that when a loop is closed the estimates can considerably change and, therefore, the tree must be rebuilt from scratch. This takes $O(n \log n)$, compatible with the cost of state recovery at loop closure.

C. *Quering a Tree of Poses*

Algorithm 2 provides a description of the implementation of a query for poses close to the current one using the tree

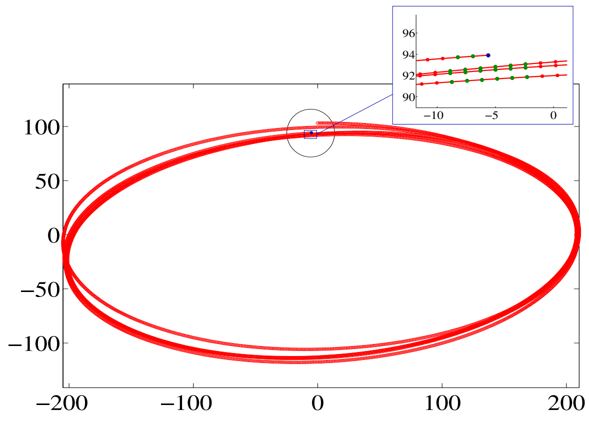

Fig. 3. Example of a simulated trajectory used to compare time execution between linear and tree-based NN search. The zoom box shows with green spots the set of poses close to the final robot pose, marked with a blue spot.

of poses. The search proceeds in a recursive way exploring only those branches that are likely to include poses close to the current one. This is evaluated using the NEIGHBOR function that implements the procedure described in Section II using the information stored in the tree nodes obtained via the GETNODEINFO function. For internal tree nodes this information is interval-based and NEIGHBOR uses interval arithmetic as described in Section IV-A. To complete the description of the algorithm, function ROOT returns the root node of a tree, HEIGHT returns the height of a tree, and functions LEFT and RIGHT return the left and right sub-trees of a particular node, respectively. All the aforementioned functions take constant time.

At the end, the search returns the labels associated with the poses that are determined to be close to the current one. Since the tree is balanced, the search process scales with $O(\log n)$.

V. EXPERIMENTS AND RESULTS

To compare the tree-based search for neighboring poses with a linear search, we used a Matlab simulation running on a Pentium at 2.4 Gz of a robot performing several loops on an elliptical trajectory with a major axis of 400 meters and a minor axis of 200 meters starting the motion at position $(0, 100)$. The length of one loop is approximately 1000 meters and store poses at one meter intervals (see Fig. 3). The robot motion is corrupted with $5cm$ and 0.5 degrees of translational and rotational standard deviation, respectively. At the end of the simulation we search for the poses that are within $v = (3, 3, 0.25)^\top$ from the last robot pose (i.e., ± 3 meters in x and y and ± 0.25 radians in orientation) with probability higher than $s = 0.5$ in a first run and $s = 0.1$ in a second one.

Fig. 4 shows the execution time in seconds when searching for neighboring poses using linear search in comparison to the tree-based search over trajectories with increasing number of poses. The longer trajectories are simulated by looping more times around the ellipse. Since trajectories are randomly generated, the results are averaged over 10 runs. The time for linear

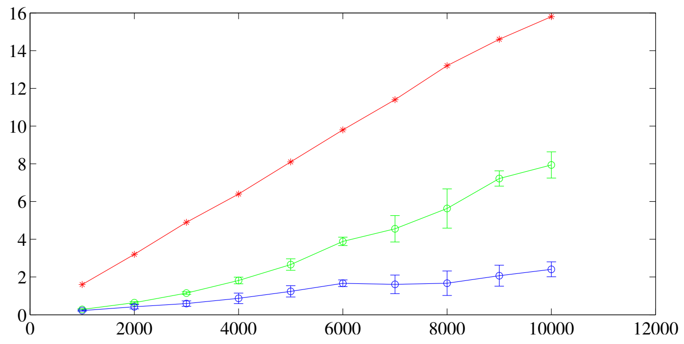

Fig. 4. Averaged execution time in seconds for 10 randomly generated experiments using a linear search (red line) and a tree-based search with $s = 0.5$ (blue line) and $s = 0.1$ (green line) for trajectories with 1000 to 10000 poses. Error bars correspond to the standard deviation.

search (red line in the plot) is the same for each execution. On the other hand the execution time for the tree-based search (blue and green lines) adapts to the structure of the trajectory and is different for each query: it is more expensive for queries where the current pose has many neighbors and faster for queries where the current pose has few neighbors. The error bars in the plot corresponds to the standard deviation of the running times. When the probability threshold s is lowered more poses pass the neighbouring test and the advantage of using a tree-based search is reduced. However, Fig. 4 shows that even when using a very low value for threshold s the tree-based search is still faster than the linear one by, at least, a factor of two. Thus, we can conclude that, as expected, the tree-based search scales more gracefully than the linear search, validating the approach introduced in this paper.

Fig. 5 shows the portion of the tree explored when solving a query in a tree with 1000 poses. White squares are internal tree nodes that pass the data association test. Red squares are tree nodes where this test fails and, consequently, where the recursive search is stopped. Yellow squares are leaves where the data association fails. Finally, green squares are the leaves returned as solution for the query. The algorithm returns a set of five poses immediately preceding the current pose in the trajectory and two poses at the beginning of the trajectory. These would be the perfect candidates for loop closure. Since those poses are stored at different parts of the tree two different branches have to be explored to determine the set of results. As shown in the figure, an nice particularity of the interval evaluation is that all branches explored lead to a valid match candidate. From those candidates, loop closure could be asserted for instance, with those poses whose distance contribute most in the revision of the network in terms of information load [13]. Note also how the technique leaves the tree balanced. For a tree with 1000 poses, the height of the tree is 11.

To validate our tree-based data association mechanism with real data, we used a data set collected at the Campus Nord of UPC in Barcelona with our service robot Tibi in the context of the URUS project. The robot is based on a Segway-RMP platform, and for this experiment, images were acquired with

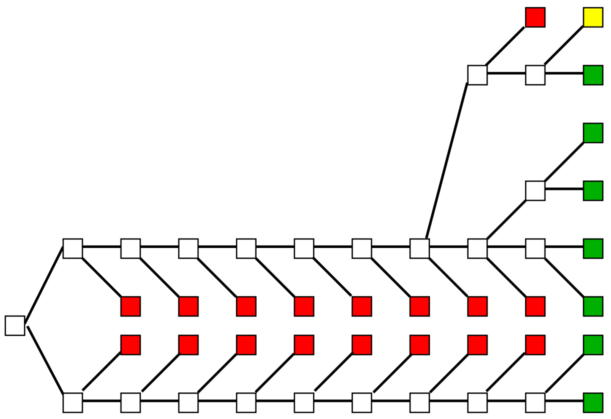

Fig. 5. Portion of the tree explored for a query with 1000 poses. White squares are internal tree nodes that pass the data association test, red squares are tree nodes where this test fails and, consequently, where the recursive search is stopped, yellow squares are leaves where the data association fails, and green squares are the leaves returned as solution for the query.

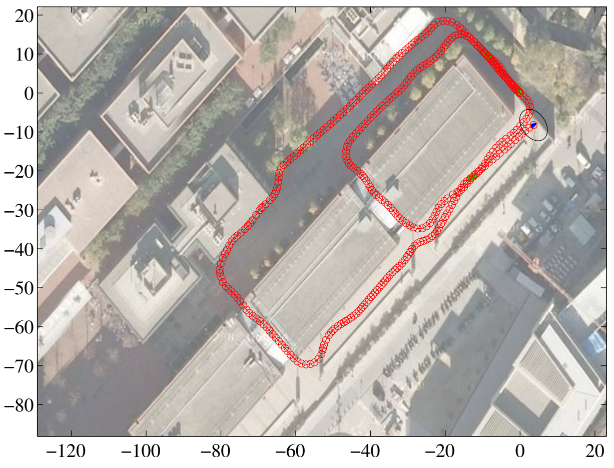

Fig. 6. Filtered trajectory using encoder and visual odometry with data collected at the Campus Nord of UPC. The trajectory includes 367 poses over 400 meters. The blue arrow indicate the final pose of the robot and the black ellipse the associated covariance at a 95% confidence level. The scale is in meters.

a Point Gray's bumblebee stereo camera. The data set includes 367 encoder-based odometry readings and stereo pair images collected over 400 meters moving around a couple of buildings in about 700 seconds (see Fig. 6). The relative displacement between pairs of stereo images is recovered from the 3d position of SIFT features [36]. This relative displacement is used both to compute visual odometry (i.e., to establish links between consecutive poses), and to assess loop closure hypotheses (i.e., to link non-consecutive poses). Sensors are modelled with constant error models with covariances large enough to comfortably cover the true error. Sensor readings are integrated using the approach described in [13] to generate a map of poses. The approach relies in an information filter and implements special procedures to form only reliable and highly informative loops. In this particular example only three loops are established, despite the trajectory overlap during long periods. The parameters for the nearest neighbor search are

	Search Method	
	Linear-based	Tree-based
Nearest Neighbor	110	83
Tree Construction	-	2
Total Filter	112	87

TABLE I
EXECUTION TIMES (IN SECONDS) FOR THE CAMPUS NORD EXPERIMENT.

	Search Method					
	Linear based	Tree based	Linear based	Tree based	Linear based	Tree based
No. of Poses	1000	1000	2000	2000	3500	3500
Nearest Neig.	1292	880	5039	2667	14353	6755
Tree Cons.	-	206	-	436	-	774
Total Filter	1750	1545	6624	4672	18759	11869

TABLE II
EXECUTION TIMES (IN SECONDS) FOR THE INTEL EXPERIMENT DISCRETIZED WITH DIFFERENT GRANULARITIES.

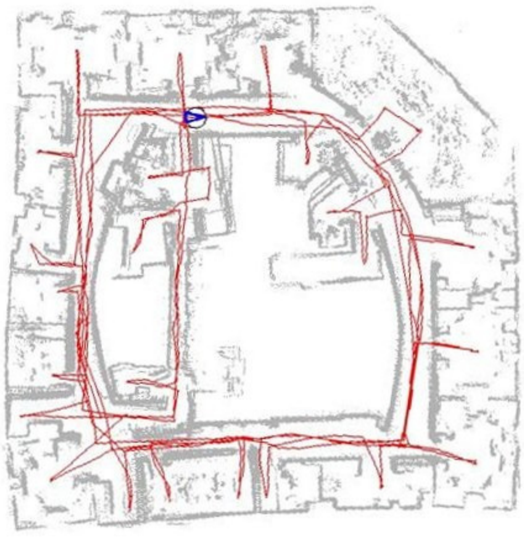
Fig. 7. Filtered trajectory using encoder and laser odometry of the Intel dataset.The blue arrow indicate the final pose of the robot and the black ellipse the associated covariance at a 95% confidence level.

$v = (2.5, 2, 0.25)$ and $s = 0.1$. The probability threshold s is purposely set this low to encourage all possible loops to be considered. Fig. 6 depicts the final trajectory as estimated by the filter. The overlay with the background image is approximate and given only as a reference.

Table I gives a comparison of the execution times in seconds for this experiment using both the linear and tree-based nearest neighbor searches. These execution times indicate only filter related processes (prediction, update, and nearest neighbor search) and do not include sensor related processes (SIFT extraction and image matching). We can see that in all cases the time devoted to nearest neighbor search clearly dominates the cost: about 95% of the time is used in this process which clearly motivates the need to improve it. With a very low probability threshold, s, many poses pass the neighboring test and the advantage of the tree-based search somehow diminishes. Despite this, the use of the tree-based approach reduces the cost of this search by a factor of 0.77. The total time needed to build and rebalance the tree is about 2 seconds only, a very small penalty to pay taking into account the computational savings obtained from using it (more than 25 seconds).

To test the performance of the tree-based data association in larger problems, we used the Intel data set from the Radish repository [37]. This data set includes about 13000 encoder-based odometry readings with their corresponding laser scans which we have used to generate laser-based odometry and to assess loop closure using an ICP scan matching algorithm [10]. After pre-processing the data set we end up with about 1000 laser scans, a database size compatible with other reported solutions for the same problem [5]. Again, for simplicity we assume error models with constant covariance matrices that underestimate the true covariances. Table II shows execution times for the filter related operations using the approach described in [13], resulting in 88 loop closures. Fig. 7 shows the estimated trajectory at the end of the process together with the raw laser scans associated with each pose. In this experiment, the cost of the search also dominant taking about a 75% of the total execution time. With parameters $v = (1, 1, 0.35)$ and $s = 0.1$ (which is a worst-case for the tree-based search) we obtain a reduction in the cost of the search with factor 0.84. Larger problems can be formed by processing the available raw sensor readings with finer granularities. If we resample the original dataset with 2000 poses instead of 1000, the reduction ratio provided by the tree-based search is 0.61. Finally, with a resample of 3500 poses the reduction is 0.52. According to the results in Fig. 4, even more substantial computational savings should be expected for larger problems.

VI. CONCLUSIONS

Pose SLAM is a variant of SLAM where only the robot trajectory is estimated. In this context, and in order to delay as much as possible filter divergence, the number of loops should be rigorously controlled, adding only the most informative links to the filter [13]. Then, the robot operates most of the time in open loop and the computational bottleneck shifts to data association and, in particular, to the search for nearest neighboring poses to be tested for sensor registration.

The estimation of the distance used for nearest neighbor search requires the joint marginals between the current robot pose and any other robot pose in the path. Previously reported techniques to compute them in exact form have at least linear complexity. In this paper we have shown that, when operating in open loop, exact joint marginals can be factorized in two independent terms that can be computed in constant time. Based on this result, we propose a technique to determine the set of neighboring poses based on a tree structure that has logarithmic computational cost.

The proposed tree-based structure evaluates the probability of the current pose being close to a subset of poses in the robot trajectory with the aid of interval arithmetic. The presented experiments both with simulated and real data and with standard datasets show that interval arithmetic offers enough accuracy to correctly identify nearby poses after long periods of open loop traverse.

The solution presented in this paper reduces the computational complexity when the robot operates in open loop despite the use of very simple error models that produce underconfident covariance estimations. The use of these error models is a disadvantageous situation for the presented search technique. In the future, we plan to derive more accurate sensors models to fully exploit the potential of the tree-based search. Finally, an aspect also deserving further attention is that of controlling the size of the map. In Pose SLAM the size of the map scales with the length of the trajectory and not with size of the area to be mapped. A control of the map size either by controlling state augmentation, or marginalizing out robot poses would be necessary to achieve a long lasting Pose SLAM system operating on a large scale environment.

Acknowledgment

This work has been partially supported by the Spanish Ministry of Science and Innovation under a Juan de la Cierva Postdoctoral Fellowship to V. Ila and the projects DPI-2007-60858, DPI-2008-06022, MIPRCV Consolider-Ingenio 2010, and the EU URUS project IST-FP6-STREP-045062 and the PACO-PLUS project IST-FP6-IP-027657.

References

[1] R. C. Smith and P. Cheeseman, "On the representation and estimation of spatial uncertainty," *Int. J. Robot. Res.*, vol. 5, no. 4, pp. 56–68, 1986.

[2] S. Thrun, Y. Liu, D. Koller, A. Y. Ng, Z. Ghahramani, and H. Durrant-Whyte, "Simultaneous localization and mapping with sparse extended information filters," *Int. J. Robot. Res.*, vol. 23, no. 7-8, pp. 693–716, Jul. 2004.

[3] M. Montemerlo and S. Thrun, *FastSLAM: A Scalable Method for the Simultaneous Localization and Mapping Problem in Robotics*, ser. Springer Tracts in Advanced Robotics. Springer, 2007, vol. 27.

[4] F. Dellaert and M. Kaess, "Square root SAM: Simultaneous localization and mapping via square root information smoothing," *Int. J. Robot. Res.*, vol. 25, no. 12, pp. 1181–1204, 2006.

[5] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Incremental smoothing and mapping," *IEEE Trans. Robot.*, vol. 24, no. 6, pp. 1365–1378, 2008.

[6] M. R. Walter, R. M. Eustice, and J. J. Leonard, "Exactly sparse extended information filters for feature-based SLAM," *Int. J. Robot. Res.*, vol. 26, no. 4, pp. 335–359, 2007.

[7] Z. Wang, S. Huang, and G. Dissanayake, "D-SLAM: A decoupled solution to simultaneous localization and mapping," *Int. J. Robot. Res.*, vol. 26, no. 2, pp. 187–204, 2007.

[8] R. Eustice, H. Singh, and J. Leonard, "Exactly sparse delayed-state filters for view-based SLAM," *IEEE Trans. Robot.*, vol. 22, no. 6, pp. 1100–1114, Dec. 2006.

[9] K. Konolige and M. Agrawal, "FrameSLAM: from bundle adjustment to realtime visual mapping," *IEEE Trans. Robot.*, vol. 24, no. 5, pp. 1066–1077, 2008.

[10] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *Auton. Robot.*, vol. 4, no. 4, pp. 333–349, 1997.

[11] U. Frese, P. Larsson, and T. Duckett, "A multigrid algorithm for simultaneous localization and mapping," *IEEE Trans. Robot.*, vol. 21, no. 2, pp. 1–12, 2005.

[12] E. Olson, J. Leonard, and S. Teller, "Fast iterative optimization of pose graphs with poor initial estimates," in *Proc. IEEE Int. Conf. Robot. Automat.*, Rome, Apr. 2007, pp. 2262–2269.

[13] V. Ila, J. Andrade-Cetto, R. Valencia, and A. Sanfeliu, "Vision-based loop closing for delayed state robot mapping," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, San Diego, Nov. 2007, pp. 3892–3897.

[14] S. J. Julier and J. K. Uhlmann, "A counter example to the theory of simultaneous localization and map building," in *Proc. IEEE Int. Conf. Robot. Automat.*, Seoul, May 2001, pp. 4238–4243.

[15] T. Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot, "Consistency of the EKF-SLAM algorithm," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Beijing, Oct. 2006, pp. 3562–3568.

[16] G. Dissanayake, S. B. Williams, H. Durrant-Whyte, and T. Bailey, "Map management for efficient simultaneous localization and mapping (SLAM)," *Auton. Robot.*, vol. 12, no. 3, pp. 267–286, May 2002.

[17] W. Zhou, J. Miro, and G. Dissanayake, "Information efficient 3D visual SLAM in unstructured domains," in *Proc. 3rd Int. Conf. Intell. Sensors, Sensor Networks and Information Process.*, Melbourne, Dec. 2007, pp. 323–328.

[18] K. Konolige, M. Agrawal, and J. Solà, "Large scale visual odometry for rough terrain," in *Proc. 13th Int. Sym. Robot. Res.*, Hiroshima, Nov. 2007.

[19] V. Ila, J. Andrade-Cetto, and A. Sanfeliu, "Outdoor delayed-state visually augmented odometry," in *Proc. 6th IFAC/EURON Sym. Intell. Auton. Vehicles*, Toulouse, Sep. 2007.

[20] I. Esteban, O. Booij, Z. Zivckovic, and B. Kröse., "SLAM for extremely large environments," in *Proc. 14th Annual Conf. Adv. School Comput. Imag.*, Heijen, Jun. 2008.

[21] K. L. Ho and P. Newman, "Detecting loop closure with scene sequences," *Int. J. Comput. Vision*, vol. 74, no. 3, pp. 261–286, Sep. 2007.

[22] R. Eustice, H. Singh, J. Leonard, and M. Walter, "Visually mapping the RMS Titanic: Conservative covariance estimates for SLAM information flters," *Int. J. Robot. Res.*, vol. 25, no. 12, pp. 1223–1242, 2006.

[23] S. Huang, Z. Wang, and G. Dissanayake, "Exact state and covariance sub-matrix recovery for submap based sparse EIF SLAM algorithm," in *Proc. IEEE Int. Conf. Robot. Automat.*, Pasadena, Apr. 2008, pp. 1868–1873.

[24] J. Neira and J. D. Tardós, "Data association in stochastic mapping using the joint compatibility test," *IEEE Trans. Robot. Automat.*, vol. 17, no. 6, pp. 890–897, Dec. 2001.

[25] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Fast incremental smoothing and mapping with efficient data association," in *Proc. IEEE Int. Conf. Robot. Automat.*, Rome, Apr. 2007, pp. 1670–1677.

[26] Y. Chen, T. A. Davis, W. W. Hager, and S. Rajamanickam, "Algorithm 887: CHOLMOD, supernodal sparse Cholesky factorization and update/downdate," *ACM T. Math. Soft.*, vol. 35, no. 3, pp. 22:1–22:14, 2008.

[27] T. M. Cover and P. E. Hard, "Nearest neighbour pattern classification," *IEEE Trans. Inform. Theory*, vol. 13, pp. 21–27, 1968.

[28] A. Farago, T. Linder, and G. Lugosi, "Fast nearest neighbour search in disimilarity spaces," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 15, no. 9, pp. 957–963, 1993.

[29] K. Fukunaga and P. M. Narendra, "A branch and bound algorihm for computing the k-nearest neighbours," *IEEE Trans. Comput.*, vol. 24, pp. 750–753, 1975.

[30] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Comm. ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[31] A. Djouadi and E. Bouktache, "A fast-algorithm for the nearest-neighbour classifier," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, no. 3, pp. 277–282, 1997.

[32] A. Beygelzimer, S. Kakade, and J. Langford, "Cover trees for nearest neighbor," in *Proc. 23rd Int. Conf. Machine Learning*, Pittsburgh, 2006, pp. 97–104.

[33] B. Zhang and S. N. Srihari, "Fast k-nearest neighbour classification using cluster-based trees," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 26, no. 4, pp. 525–528, 2004.

[34] R. E. Moore, *Interval Analysis*. Prentice Hall, 1966.

[35] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, ser. MIT Electrical Engineering and Computer Science Series. Cambridge: MIT Press, 1992.

[36] D. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[37] A. Howard and N. Roy, "The robotics data set repository (Radish)," 2003, available: http://radish.sourceforge.net/.