Proceedings of the RAAD 2011
20th International Workshop on Robotics in Alpe-Adria-Danube Region
October 5-7, 2011, Brno, Czech Republic

A Polymorph Particle Filter for State Estimation used by a Visual Observer

Markus Bader and Markus Vincze

Automation and Control Institute (ACIN), Vienna University of Technology, Austria
E-mail: {bader, vincze}@acin.tuwien.ac.at
URL: http://www.acin.tuwien.ac.at

Abstract. This paper presents a novel particle filter that is not only able to estimate the state but also capable of estimating the current best model to describe the state.
Particle filters as well as Kalman filters are a common technique to track and predict the state of a system. The requirement for both filters is knowledge of the underlying system to correctly apply a process model. However, different states typically require different models. This work presents a particle filter that uses polymorphic particles which represent different models related to the various system states. This improves performance compared to running multiple filters in parallel. The showcase is a controlled crane. The observers' task is to initialize the real-time control of the cranes' anchor by predicting the current systems' state, and to plan collision-free paths to avoid detected obstacles. The possible states of the anchor are: (a) stands still, (b) swings freely or (c) is under control. Our new particle filter votes are based on the best fitting particles for a certain model and inject particles to boost its performance.

Keywords. Robotic, Visual Servo Control, Computer Vision, Particle Filter

1. Introduction

Direct vision-based concepts, also known as visual servo controls, have been researched for many years. A well written introduction can be found at (Chaumette and Hutchinson, 2006). But outside of the automation industry, only a few vision-based systems have been implemented, especially since humans can be endangered by a system fault. Car lane departure warning systems[1], where the system "softly" interacts in real-time if a car leaves the driving lane, is one of the few examples that reached the market.
One of the possible reasons for this could be that such assistance has to adapt by recognizing varying behaviour of the operator. And exactly this switching of behaviour can be detected and estimated by the newly proposed polymorphic particles in addition to the systems state.
Our showcase presents an overhead crane, "see Fig. 1", which can be controlled by hand or by a very sophisticated control, and a path-planning program proposed and implemented at our institute (Graichen et al., 2010, Egretzberger et al., 2010).

This crane should be visually observed to estimate its state and to give feedback if the control is not working as expected. The possible states of the system to be observed are if the anchor/hook: (a) stands still, (b) swings freely or (c) is under control. The proposed filter injects particles with different motion models (i) random motion, (ii) 3D pendulum motion or (iii) a linear motion related to the three different states. The winning particle type then presents the best approximated motion and the systems state.

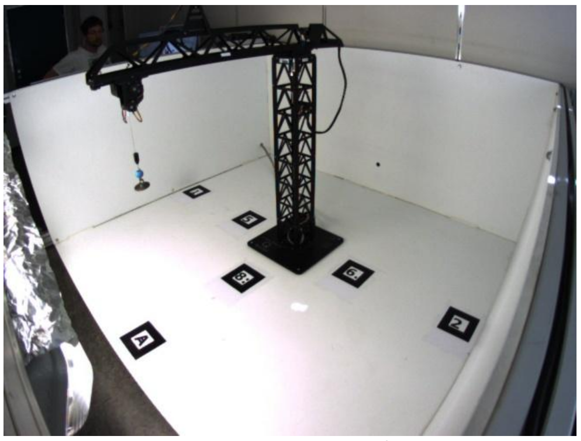


Fig. 1. Overhead crane used as test platform

[1] Lane departure warning system:
http://en.wikipedia.org/wiki/Lane_departure_warning_system

This enables us to predict the crane's hook trajectory to reinitialize the system if needed.
This document is structured as follows: first, we present the currently most related work. After this, the setup will be presented to demonstrate the filter. Section 5 presents the proposed filter, followed by results and a brief conclusion.

2. Related Work

Different projects have dealt with the control of an overhead crane. Relevant work on the control of a crane using visual input was previously carried out (Usher et al., 2005). But the cable array robot presented was, unlike our robot, able to control all degrees of freedom (DOF). Generally we can split the related work into two topics:

- Visual servo control
- State estimation using particle filters

2.1. Visual servo control

Typical visual servo controls are designed to minimize an error $e(t)$ which is generally described in (Chaumette and Hutchinson, 2006) with Eq. (1).

$$e(t) = s(m(t), a) - s^* \qquad (1)$$

s^* denotes the desired control pose and s denotes the current estimated pose based on visual features $m(t)$ and on the additional knowledge a on the system. Chaumette presented in (Chaumette and Hutchinson, 2007) how the epipolar geometry can be used as additional knowledge to improve the control.
(Chaumette and Hutchinson, 2006) also pointed out that there are two major classes in visual servoing: image-based visual servo control (IBVS) and position-based visual servo control (PBVS). Since we are using multiple cameras to estimate the position of our object of interest, we are dealing with PBVS in this paper. s related to Eq. (1) consists therefore of 3D parameters.

2.1. State estimation using particle filters

Particle filters as well as Kalman filter / extended Kalman filters (EKF) are common techniques to track and predict the state of a system. But typical EKF are bound to unimodal distributions while particle filters are known to be non-parametric and able to deal with unforeseen events like kidnapping problems (Thrun et al. 2005).
Particle filters are also common on tracking objects in images (Nummiaro et al. 2003). Nummiaro presented how color features can be used to design an object tracker based on a particle filter which is more robust against partial occlusion, rotation and scale invariant than other Kalman filters at this time.

3. Setup

The semi autonomous crane model, which is used, is able to pick up and place objects.
Fig. 2 shows a schematic block view of the system involving the cycle times and transport protocols between components.

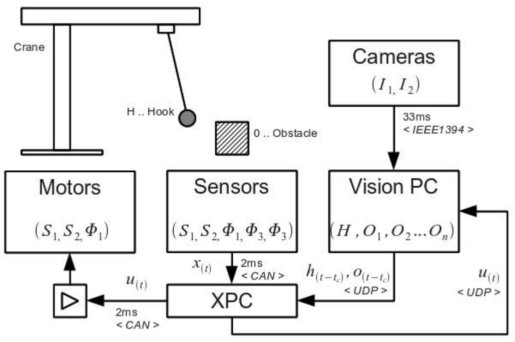


Fig. 2. Schematic system overview

3.1. Low Level Controller

The low-level control of the crane is based on relative encoders mounted on all joints related to the 5 DOFs $[s_1, s_2, \phi_1, \phi_2, \phi_3]^T$. Trolley position s_1, cable length s_2, jip rotation ϕ_1 and the two actuated gimbal angles (ϕ_2, ϕ_3) are related to the cable origin.
The control is able to cope with disturbances and tries to keep the crane's hook on the planned trajectory. Additional constraints on the implemented model predictive controller (MPC) (Graichen and Kugi 2010) can be set to compute even energy-efficient trajectories.
The implementation is done on a real-time XPC Target using Matlab and runs with a constant cycle of 5ms. No visual information is used within this control cycle to keep the hook on track.
The relative encoders are forcing the system to be initialized from a known position. To overcome this problem, the vision system should provide the control with the observed system's state to enable itself to reinitialize.

3.2 Vision System

The designed vision system is able to use multiple cameras but only two cameras are used in the setup.
The camera pose relative to the setup is computed online using visual markers[2], which are also called *fiducials*. This enables us to work even with moving cameras.

[2] ARToolKit: http://www.hitl.washington.edu/artoolkit/

The visual detection of the hook within the camera image can be seen as $m(t)$ Eq. (1) and is based on a common colour blob detection algorithm (Bruce et al., 2000). Unlike other systems, our system is designed to work with unsynchronized cameras which on one the hand pose additional problems shown in "see Fig. 3", but on the other hand it allows us to extend the setup easily over a cloud of cameras. How to estimate the 3D location of a point P seen in two cameras images at the same time is described in (Richard H. and Andrew Z. 2004). But our system has to deal with unsynchronized cameras. Therefore, we have to estimate the motion of P additionally.

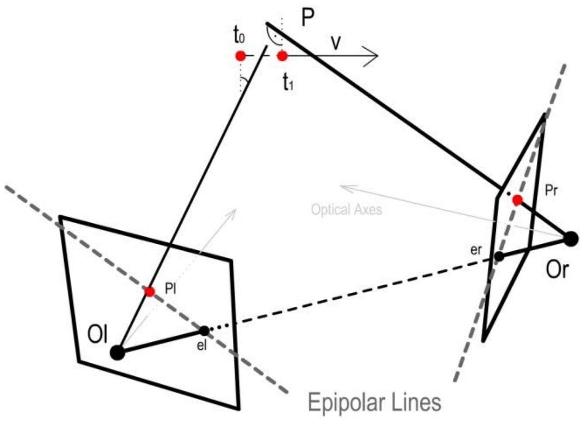


Fig. 3. Triangulation problem with unsynchronized cameras. The point P moves between the two camera snapshots

It is obvious that the key to realizing such a system lies in the accurate visual detection but it is also necessary to predict the hook's trajectory in order to overcome the system latencies due to the image processing and the time needed to share the information between the low-level control and the vision system.

3.3 Framework

The Framework is partially based on ROS (Quigley et al., 2009). ROS uses RPC calls to connect programs called notes[3]. The implemented vision pipeline is a collection of multiple notes, which can be distributed over multiple computers. ROS in our case connects:

- two camera driver notes
- two image distortion notes
- two marker detection notes
- two colour blob detection notes
- one particle filter and
- one UDP bridge note

together.

The UDP bridge is needed to communicate with the XPC. The disadvantage of this system lies in the signal driven communication, which has at best soft-real-time capabilities.
This leads to a delay of at least 100ms, while sending processed vision information back to the low-level control.

4. Particle Filter

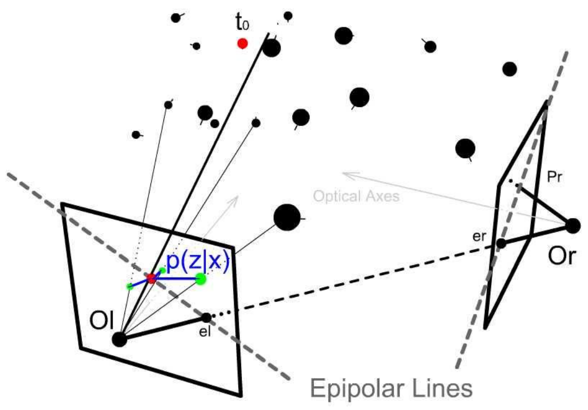
Sharing data between components needs time which causes data to be outdated by the time it reaches its destination. A carefully designed filter can describe the system's state in such a way that an estimate of system's state can be realized for the time instance in the future, which can prevent outdated information caused by communication latencies.
This Section describes now the implemented filter in detail, which is able to solve the following previously stated problems:

- 3D triangulation of a moving object seen in unsynchronized camera images.
- Model selections to describe the hooks´ motion to enable pose predictions.

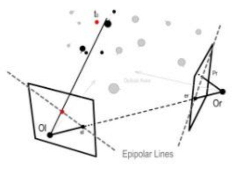
3.1. 3D triangulation

To simplify this, we will describe the filter by using a linear motion model. The polymeric properties of our particles will be described in the next section.
The first steps of our particle filter are implemented like a monte carlo particle filter (MCL) described in (Thrun et al. 2005).
The selected importance factor $p(z \mid x)$ of each particle is based on the particles projection onto the image plane "see Fig. 4a." where we assumed a Gaussian error to the nearest detected colour blob.
x denotes the location of the particle in 3D space and z the nearest detected blob on the image plane.
All particles are sorted by their importance factor.
σ_{best} defines a threshold related to the total number of particles which we are preventing from being resampled.
Next we are going to resample all the particles which not under our σ_{best} into the one in our σ_{best} limit "see Fig. 4b"
A motion update with an additional noise will be performed for each particle accordingly to its assigned motion model, "see Fig. 4c".
Fig. 4d-e shows how the particles are accumulated around the real object position and motion.

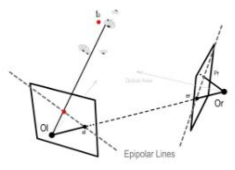
[3] Robot Operating System (ROS): http://www.ros.org/



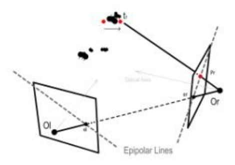
a.) Uniform distributed particles and importance factor measured on the image plane



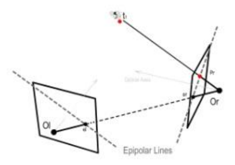
a) σ_{best} particles



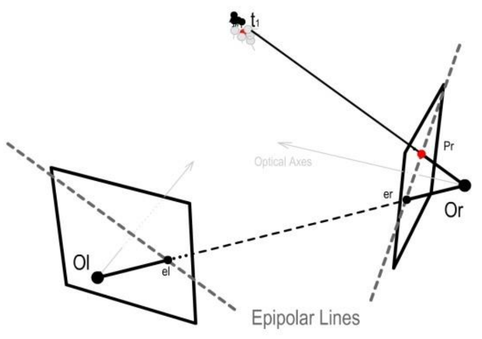
b) Resample



c) Update



d) σ_{best} particles



e) Triangulated particles

Fig. 4. Particle Filter triangulation

3.2. Polymorphic Particles

If we are looking from an object-oriented programming site onto our particles, we see that every particle must provide certain functionality. A particle must be able to:

- place itself on a random location picked from a *uniform* distribution over the state space limited by a region of interest (ROI) .
- move itself during the *update* accordingly to its motion and noise model.
- copy the motion parameter of another particle to *resample* itself

Based on this idea, we designed a filter written in C++ where the particles are implemented using classical object-oriented strategies. This strategy allows us to generate particles that look identical to each other but act differently. Such techniques are known by the term "polymorphism" (Stroustrup B., 2000). This allows us to sample particles of different types over a ROI. If we perform the steps through our filter, a certain type of particle will converge around our real state.

The downside here is that we have to prevent particle types from becoming extinct and we have to introduce an additional function that initializes the motion model behind a particle based on another one. This means every particle needs a function to:

- initialize itself based on a particle with a different motion model to *inject* a endangered particle type.

Fig. 5 visualizes the classes by using a UML diagram.

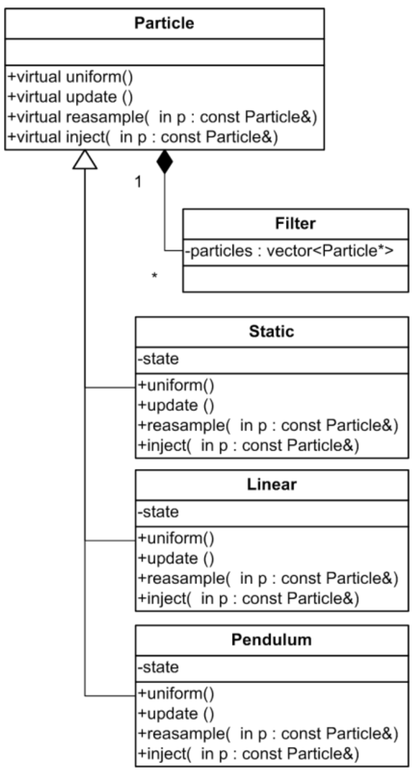

Particle
+virtual uniform()
+virtual update ()
+virtual reasample(in p : const Particle&)
+virtual inject(in p : const Particle&)
1
Filter
-particles : vector<Particle*>
*
Static
-state
+uniform()
+update ()
+reasample(in p : const Particle&)
+inject(in p : const Particle&)
Linear
-state
+uniform()
+update ()
+reasample(in p : const Particle&)
+inject(in p : const Particle&)
Pendulum
-state
+uniform()
+update ()
+reasample(in p : const Particle&)
+inject(in p : const Particle&)

Fig. 5. Class diagram of the implemented polymorphic particle with the relation to the filter class and the virtual function in our base class.

As an example, we would like to show how 2D linear motion can be used on our crane to initialize a particle with a 2D pendulum motion, where the height of the crane is known h_{crane} .

A pendulum motion can be described by Eq. (2).

$$\Theta_t = \Theta_{\max} \cos\left(t\sqrt{g/l}\right) \quad (2)$$

Fig. 6 shows how particles with a pendulum motion can be initialized by a particle with a linear motion.
l can be computed by using the normal on the motion vector v. $\Theta_{\max}$ depends on the kinetic and potential energy Eq. (3-4).

$$\Theta_{\max} = \arccos\left(1 - \frac{F}{mg}\right) \quad (3)$$

$$F = F_k + F_p = \frac{1}{2v^2} + mgl\,(1 - \cos(\Theta)) \quad (4)$$

This allows us to use motion models that would never converge if used in a standard filter. Because motion models like the pendulum motion are based on more than eight variables, the changes caused by fitting particles at the right location out of a uniform distribution would fail.
Our approach on the other hand will inject particles of a higher order just into the centre of particle clouds of a lower order. These increase the chances of fitting particles with higher state order which then are able to converge.

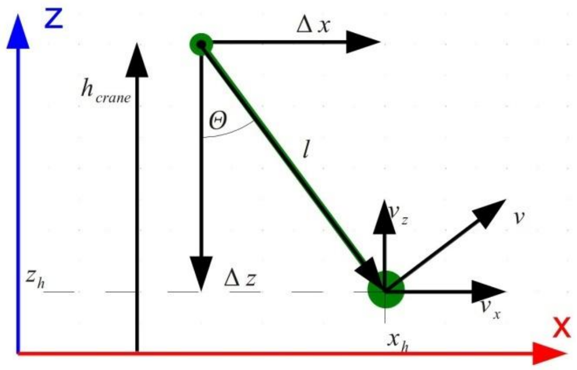


Fig. 6. Initialization of a particle with a pendulum motion from a linear motion.

5. Results

The results on our setup showed that our particle filter is able to converge to the real state and to select between different motion models if needed.
Fig. 7 shows that the filter was able to switch from a static model with random noise *red* to a linear motion model *blue*.
Tests also showed that the filter is able to switch to a 3D pendulum model if the hook swings free without control, “see Fig. 8”.

Due to the better motion model we were also able to reduce the number of particles that lead to a lower computational overhead.

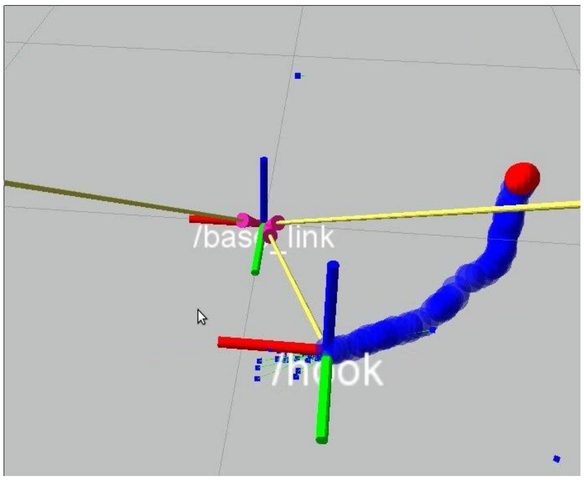


Fig. 7. History of estimates and predictions in front of the hook. Red spheres mark states where particles of static motion models were dominant, and blue where linear motion particles were winning.

The triangulation problem was solved within the filter and we were able to cope with occlusions as long as the motion model was estimated well enough to cope with the loss of information.
The system can now be initialized from arbitrary positions. Due to latencies, we are still trying to estimate all parameters needed to initialize the crane while the hook is swinging freely.

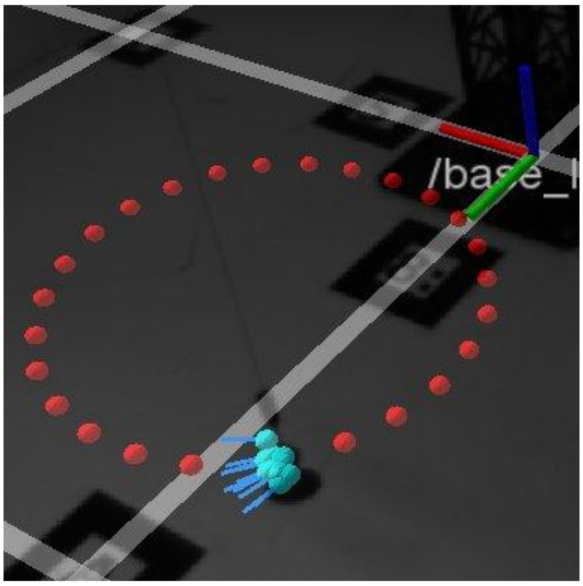


Fig. 8. Circular pendulum motion approximated by the filter.

6. Conclusion

We presented a polymorphic particle filter that is able to distribute particles over a region of interest and over a selection of different motion models.
An intelligent reinjection of endangered motion models ensures that the best motion models to approximate the systems state converge.
The use of motion models with a higher state order enabled us to use less particles which lead to lower computational costs.
Based on the result of the filter we were able to identify when the crane stood still and we were also able to reinitialize it which prevented the relative encoders to drift.
In the future, we are going to use the filter on mobile robots. This would allow us to estimate not only the position of the robot but would also give us an idea which task should be executed.

7. References

Chaumette F. and Hutchinson S. 2006 Visual servo control. I. Basic approaches, *Robotics Automation Magazine, IEEE*, Vol. **13**, pp.82-90.

Chaumette F. and Hutchinson S. 2007 Visual servo control. II. Advanced approaches [Tutorial], *Robotics Automation Magazine, IEEE*, Vol. **14**, pp.109-118.

Graichen K., Egretzberger M. and Kugi A. 2010 Suboptimal model predictive control of a laboratory crane. 8th *IFAC Symposium on Nonlinear Control Systems*, pp. 397-402.

Egretzberger M. , Graichen K. and Kugi A. 2010 Flatness-based MPC and global path planning towards cognition-supported pick-and-place tasks of tower cranes. *Advanced Dynamics and Model-Based Control of Structures and Machines*

Graichen K. and Kugi A. 2010 Ein suboptimaler Ansatz zur schnellen modellprädiktiven Regelung nichtlinearer Systeme. *at - Automatisierungstechnik*, Vol. **58**, No.8 pp.447-456.

Bruce J., Bruce J., Balch T. and Veloso M. 2000 Fast and inexpensive color image segmentation for interactive robots. *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000).*

Quigley M., Conley K., Gerkey B., Faust J., Foote T., Leibs J., Wheeler R. and Y. Ng. A. 2009 ROS: an open-source Robot Operating System. *ICRA Workshop on Open Source Software*.

Thrun S., and Burgard W., and Fox D. 2005 Probabilistic Robotics (Intelligent Robotics and Autonomous Agents). *The MIT Press.*

Stroustrup B. 2005 The C++ programming language. *O'Reilly Media, Inc. ISBN 0-201-70073-5.*

Usher K., Winstanley G. J., Corke P. I., Stauffacher D. and Carnie R. 2005, Air vehicle simulator: an application for a cable array robot. *ICRA 2005. Proceedings of the 2005 IEEE International Conference on,* pp. 2241-2246

Nummiaro K., Koller-Meier E. and Gool L.V. 2003, An Adaptive Color-Based Particle Filter, 2005. *Image and Vision Computing,* Vol. **21**, No.1 pp. 99-110.