# A High-Speed Object Tracker from Off-the-Shelf Components

Christoph H. Lampert, Jan Peters

Max Planck Institute for Biological Cybernetics, Tübingen, Germany

MAX-PLANCK-GESELLSCHAFT

BIOLOGISCHE KYBERNETIK

## Overview

**We introduce RTblob, an open-source real-time vision system for 3D object detection that achieves over 200 Hz tracking speed with only off-the-shelf hardware component. It allows fast and accurate tracking of colored objects in 3D without expensive and often custom-built hardware, instead making use of the PC graphics cards for the necessary image processing operations.**

## 1 Background

For many interesting robotics applications it is necessary to process the visual input of one or more cameras is real-time, in order to allow interaction between the robot and its environment, e.g. for *visual servoing*, *basketball dribbling*, or *catching balls*. Previous solutions for tracking object in cameras images either require markers on the object (*e.g.* VICON setups), or they are not capable of running at high framerate of 100 Hz or more (*e.g.* TYZX, *OpenCV*). Fast markerless systems have been developed, but they are too expensive for most robotics research projects, because they rely on specialized hardware, such as FPGAs [1, 2, 3] or even custom-built chips [4].

To overcome these limitation, we introduce **RTblob**, a vision system for tracking simple colored objects that is at the same time **very fast**, runs on **inexpensive hardware**, and is available free of charge in **open source** form. RTblob uses a modular feed-forward architecture that makes it simple to use also for non-experts, and that allows easy adaption *e.g.*, to different camera setups and tracking tasks.

**RTblob source code is available at `http://www.rtblob.de`.**

## 2 Object Detection Pipeline

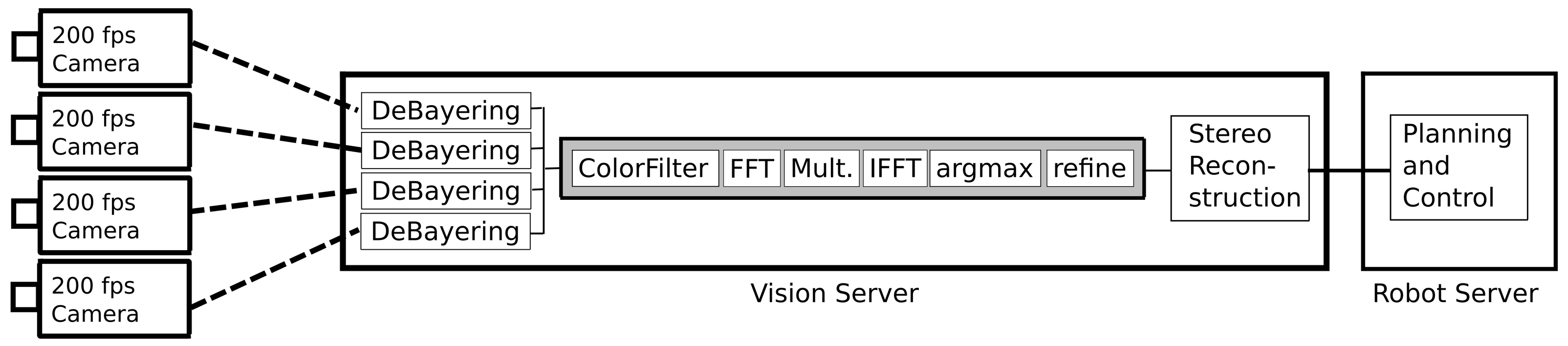
Many existing object detection algorithms are either too slow or too fragile for use in a real robotics system. To achieve maximal speed and simplicity, we use a well-understood object detection pipeline based on linear shift invariant filters [5]. Using the NVIDIA CUDA framework, we are able to parallelize the computation between CPU and GPU, thereby achieving minimal latency and maximal throughput.

### 1) Transfer Image Data from Cameras to CPU

This required a connection with high constant transfer rate (60 MB/s for each $640 \times 480$ camera at 200 fps). We rely on Gigabit Ethernet with DMA-enabled drivers, but RTblob could also be adapted to FireWire or USB 2.0.

### 2) Debayering

Most high-framerate cameras capture images in raw Bayer format, where the Bayer pattern can differ between manufactorer. We perform debayering on the CPU using the manufacturer's SDK, thereby staying flexible in choice of the camera models.




**Example Setup: 200 Hz Four Camera System**

### 3) Image Transfer: CPU to GPU

We copy the debayered image to the graphics card for further processing. *PCI Express* graphics cards can easily handle the necessary memory transfer bandwidth of 175 MB/s per camera.

### 4) Color Filtering

To search for an object of a specified color, we calculate an *interest image*:

$$I[u,v] = 1 - \frac{1}{3}\Big((\alpha_R R^\gamma[u,v] - R_{ref})^{1/\gamma} + (\alpha_G G^\gamma[u,v] - G_{ref})^{1/\gamma} + (\alpha_B B^\gamma[u,v] - B_{ref})^{1/\gamma}\Big),$$

with camera dependent gains $(\alpha_R, \alpha_G, \alpha_B)$ and gamma correction $\gamma$. Such color filtering is a per-pixel operation and can be performed very fast on the GPU by starting one thread per pixel. The filter mask is problem specific. For colored balls, we use a slightly elliptic variant of a *Laplacian of Gaussians* [6] filter.

### 5) Object Detection my Maximizing a Linear Filter Response

Object detection consists of maximizing the response of a linear shift invariant filter:

$$[\hat{u}, \hat{v}] = \operatorname{argmax}_{u,v}\ F_{u,v} * I$$

which can be calculated efficiently by using the *convolution theorem* [7]:

$$= \operatorname{argmax}_{u,v}\ \mathcal{F}^{-1}\left(\mathcal{F}(F) \cdot \mathcal{F}(I)\right)[u,v] \qquad (1)$$

where $\mathcal{F}$ denotes the Fourier transform. FFTs can be calculated very fast on the GPU. We rely on a CUFFT library by NVIDIA. Similarly, there are efficient ways to quickly compute the argmax operation. Overall, computing (1) takes less than 1 ms on a modern graphics card.

### 6) Sub-Pixel Refinement

The solution $[\hat{u}, \hat{v}]$ to (1) lies on a grid with single pixel resolution. We refine this to subpixel accuracy using a quadratic surface interpolation of the filter response [8].

### 7) Transfer Object Coordinates: GPU to CPU

Transferring the detected 2D object position back to the CPU's main memory is of negligible effort, because it consists of only a few bytes of data.

### 8) Triangulation

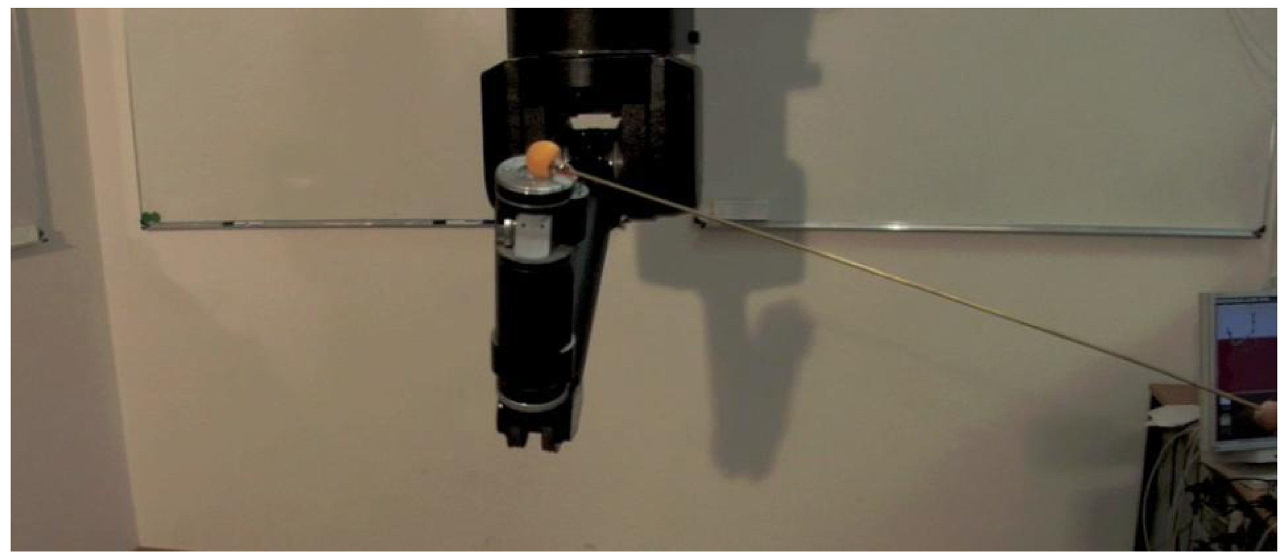
After having the 2D object positions from all cameras available, we use standard stereo triangulation [9] to estimate the object's position in 3D world coordinates.

## 3 Performance Measurements

| # of cameras | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| CPU load | $\approx$ 58% | $\approx$ 62% | $\approx$ 75% | $\approx$ 78% | $\approx$ 80% |
| detection speed | – | 841 $\pm$ 4 Hz | 421 $\pm$ 9 Hz | 272 $\pm$ 3 Hz | 209 $\pm$ 4 Hz |

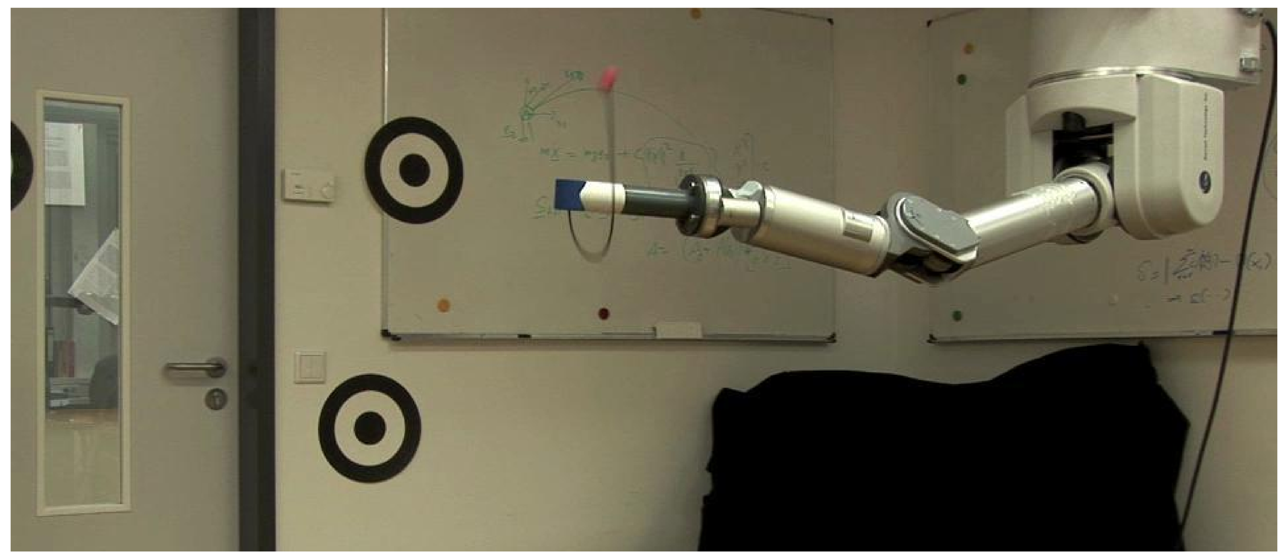
(Hardware plattform: Dell Inspiron T7400 with 3.4 GHz Intel quadcore CPU, NVIDIA GeForce GTX 280, Intel PRO/1000 PT Quad Port gigabit Ethernet card, 4 Prosilica GE640C cameras.)

## 4 Example Applications



**Visual Servoing**

**Striking a Table Tennis Ball**



**Kendarma (Ball-in-a-Cup)**

## References

[1] U. Mühlmann, M. Ribo, P. Lang, and A. Pinz, "A new high speed CMOS camera for real-time tracking applications," in *ICRA*, 2004.

[2] K. Shimizu and S. Hirai, "CMOS+FPGA vision system for visual feedback of mechanical systems," in *ICRA*, 2006.

[3] Y. Watanabe, T. Komuro, and M. Ishikawa, "955-fps real-time shape measurement of a moving/deforming object using high-speed vision for numerous-point analysis," in *ICRA*, 2007.

[4] Y. Nakabo, M. Ishikawa, H. Toyoda, and S. Mizuno, "1ms column parallel vision system and its application of high speed target tracking," in *ICRA*, 2000.

[5] B. Jähne, *Digital image processing: concepts, algorithms, and scientific applications.* Springer Berlin, 1995.

[6] T. Lindeberg, *Scale-Space Theory in Computer Vision.* Kluwer Academic Publishers Norwell, MA, USA, 1994.

[7] J. Lewis, "Fast template matching," in *Vision Interface*, 1995.

[8] Q. Tian and M. N. Huhns, "Algorithms for subpixel registration," *Computer Vision, Graphics, and Image Processing*, 1986.

[9] R. Hartley and A. Zisserman, *Multiple view geometry.* Cambridge University Press, 2000.