Adaptive Probabilistic Tracking Embedded in a Smart Camera

Sven Fleck and Wolfgang Straßer
WSI/GRIS, University of Tübingen
{fleck,strasser}@gris.uni-tuebingen.de

Abstract

Tracking applications based on distributed and embedded sensor networks are emerging today, both in the field of surveillance (airports, lab facilities, train stations, museums, public spots) and industrial vision (visual servoing, factory automation). Traditional centralized approaches offer several drawbacks, due to limited communication bandwidth, computational requirements and thus also limited spatial camera resolution and framerate.

In this paper, we present a network-enabled Smart Camera for probabilistic tracking. It is capable of tracking objects adaptively in real-time and offers a very bandwidth-conservative approach, as the whole computation is performed embedded in the Smart Camera, and only the tracking results are transmitted which are on a higher level of abstraction.

1 Introduction

In typical computer vision systems today, cameras are seen only as simple sensors. The processing is performed after transmitting the complete raw sensor stream via a costly and often distance-limited connection to a centralized processing unit (PC). We think it is more natural to also embed the processing in the camera itself: what algorithmically belongs to the camera is also physically performed in the camera. The idea is to compute the information where it becomes available – directly at the sensor – and transmit only results that are on a higher level of abstraction. This follows the emerging trend of self contained and networking capable Smart Cameras.

We present a network-enabled Smart Camera capable of embedded probabilistic object tracking in real-time. Tracking plays a central role for many applications including robotics (visual servoing, RoboCup), surveillance (person tracking) and also human-machine interface, motion capture, augmented reality and 3DTV. Due to the embedded and decentralized nature of this vision system, besides real-time constraints the robust and fully autonomous operation is an essential challenge, as no user interaction is available during the tracking operation. This is achieved by various concepts: First, using particle filtering techniques enables the robust handling of multimodal pdfs and nonlinear systems. Second, an adaptivity mechanism increases the robustness by adapting to slow appearance changes of the target. The third component is a simple diagnosis unit that controls the degree of adaptivity. It is based on a confidence measure of the tracker.

Particle filters have become a major way of tracking objects [1, 2, 3]. Utilized visual cues include shape [3] and color [4, 5, 6, 7] or a fusion of cues [8, 9]. The adaptive particle filter algorithm is described in section 2. We use a color histogram based approach adapted to the special needs of our hardware target. Our Smart Camera tracking architecture is described in section 3. Afterwards, we discuss various benefits of our approach and show experimental results in section 4 before we conclude this paper.

2 Particle Filter

Particle filters can handle multiple hypotheses and nonlinear systems. Following the notation of Isard and Blake [3], we define $Z_t$ as representing all observations $\{z_1, ..., z_t\}$ up to time t, while $X_t$ describes the state vector at time t with dimension k. Particle filtering is based on the Bayes rule to obtain a posterior $p(X_t|Z_t)$ at each time-step using all available information:

$$p(X_t|Z_t) = \frac{p(z_t|X_t)p(X_t|Z_{t-1})}{p(z_t)} \quad (1)$$

whereas this equation is evaluated recursively as described below. The fundamental idea of particle filtering is to approximate the probability density function (pdf) over $X_t$ by a weighted sample set $S_t$. Each sample $s_i$ consists of the state vector $X$ and a weight $\pi$, with $\sum_{i=1}^{N} \pi^{(i)} = 1$. Thus, the i-th sample at time t is denoted by $s_{i}^{(t)} = (X_{i}^{(t)}, \pi_{i}^{(t)})$. Together they form the sample set $S_t = \{s_{i}^{(t)} | i = 1, ..., N\}$.

Fig. 1 shows the principal operation of a particle filter with 8 particles, whereas its steps are outlined below.
Choose Samples Step: First, a cumulative histogram of all samples’ weights is computed. Then, according to each particle’s weight \( \pi^{i}_{t-1} \), its number of successors is determined according to its relative probability in this cumulative histogram.

Prediction Step: In the prediction step, the new state \( X_t \) is computed:

\[
p(X_t|Z_{t-1}) = \int p(X_t|X_{t-1})p(X_{t-1}|Z_{t-1})dX_{t-1}
\]

Different motion models are possible to implement \( p(X_t|X_{t-1}) \). We use three simple motion models (whereas the specification of how many samples belong to each model can be parameterized): a random position model, a zero velocity model and a constant velocity model \((X_t = AX_{t-1} + \omega_{t-1})\), each enriched with a Gaussian diffusion \( \omega_{t-1} \) to spread the samples and to allow for target moves differing from each motion model. Our state has the form \( X^{(i)}_t = (x, y, v_x, v_y)^{(i)}_t \).

Measurement Step: In the measurement step, the new state \( X_t \) is weighted according to the new measurement \( z_t \) (i.e., according to the new sensor image).

\[
p(X_t|Z_t) = p(z_t|X_t)p(X_t|Z_{t-1})
\]

The measurement step (3) complements the prediction step (2). Together they form the Bayes formulation (1).

2.1 Color Histogram based Particle Filter

Measurement Step in context of Color Distributions

As already mentioned, we use a particle filter on color histograms. This offers rotation invariant performance and robustness against partial occlusions and non-rigidity. In contrast to using standard RGB space, we use a HSV color model: A 2D Hue-Saturation histogram \((HS)\) in conjunction with a 1D Value histogram \((V)\) is designed as representation space for (target) appearance. This induces the following specializations of the abstract measurement step described above.

From Patch to Histogram

Each sample \( s^{(i)}_t \) induces an image patch \( P^{(i)}_t \) around its spatial position in image space, whereas the patch size \((H_x, H_y)\) is user definable. To further increase the robustness of the color distribution in case of occlusion or in case of present background pixels in the patch, an importance weighting dependent on the spatial distance from the patch’s center is used. We employ the following weighting function:

\[
k(r) = \begin{cases} 
1 - r^2 & r < 1 \\
0 & \text{otherwise}
\end{cases}
\]

with \( r \) denoting the distance from the center. Utilizing this kernel leads to the color distribution for the image location of sample \( s^{(i)}_t \):

\[
p^{(i)}_t[b] = f \sum_{w \in P^{(i)}_t} k \left( \frac{\|w - \bar{X}^{(i)}_t\|}{a} \right) \delta[I(w) - b]
\]

with bin number \( b \), pixel position \( w \) on the patch, bandwidth \( a = \sqrt{H_x^2 + H_y^2} \) and normalization \( f \), whereas \( \bar{X}^{(i)}_t \) denotes the subset of \( X^{(i)}_t \) which describes the \((x, y)\) position in the image. The \( \delta \)-function assures that each summmand is assigned to the corresponding bin, determined by its image intensity \( I \), whereas \( I \) stands for \( HS \) or \( V \) respectively. The target representation is computed similarly, so a comparison to each sample can now be carried out in histogram space.

From Histogram to new Weight \( \pi \)

Now we compare the target histogram with each sample’s histogram: For this, we use the popular Bhattacharyya similarity measure [4], both on the 2D \( HS \) and the 1D \( V \) histograms respectively:

\[
\rho[p^{(i)}_t[b], q_t[b]] = \sum_{b=1}^{B} \sqrt{p^{(i)}_t[b]q_t[b]}
\]

with \( p^{(i)}_t \) and \( q_t \) denoting the \( i \)-th sample and target histograms at time \( t \) (respectively in Hue-Saturation \((HS)\) and Value \((V)\) space). Thus, the more similar a sample appears to the target, the larger \( \rho \) becomes. These two similarities
\( \rho_{HS} \) and \( \rho_V \) are then weighted using alpha blending to get a unified similarity. The number of bins is variable, as well as the weighting factor. The experiments are performed using \( 10 \times 10 + 10 = 110 \) bins (\( H \times S + V \)) and a 70 : 30 weighting between \( HS \) and \( V \). Then, the Bhattacharyya distance

\[
d_t^{(i)} = \sqrt{1 - \rho[p_t^{(i)}][b], q[b]}
\]

is computed. Finally, a Gaussian with user-definable variance \( \sigma \) is applied to receive the new observation probability for sample \( s_t^{(i)} \):

\[
\pi_t^{(i)} = \frac{1}{\sqrt{2\pi\sigma}} \exp \left( -\frac{d_t^{(i)2}}{2\sigma^2} \right)
\]

Hence, a high Bhattacharyya similarity \( \rho \) leads to a high probability weight \( \pi \) and thus the sample will be favored more in the next iteration. Fig. 2 illustrates how the variance \( \sigma^2 \) affects the mapping between \( \rho \) and the resulting weight \( \pi \). A smaller variance leads to a more aggressive behavior in that samples with higher similarities \( \rho \) are pushed more extremely.

![Figure 2. Mapping of Bhattacharyya similarity \( \rho \) to weight \( \pi \) for different variances \( \sigma^2 \)](image)

### 2.2 Self-Adaptivity

To increase the tracking robustness, the camera automatically adapts to slow appearance (e.g., illumination) changes during runtime. This is performed by blending the appearance at the most likely position with the actual target reference appearance in histogram space:

\[
q_t[b] = \alpha \times p_t^{(i)}[b] + (1 - \alpha) \times q_{t-1}[b]
\]

for all bins \( b \in \{1, ..., B\} \) (both in \( HS \) and \( V \)) using the mixture factor \( \alpha \in \{0, ..., 1\} \) and the maximum likelihood sample \( j \), i.e., \( \pi_t^{(i)} = \max_{i=1,...,N} \{\pi_t^{(i)}\} \). The rate of adaption \( \alpha \) is variable and is controlled by a diagnosis unit that measures the actual tracking confidence. The idea is to \textit{adapt wisely}, i.e., the more confident the Smart Camera is itself about actually tracking the target, the less is the risk of overlearning and the more it adapts to the actual appearance of the target. The degree of unimodality of the resulting pdf \( p(X_t|Z_t) \) is one possible interpretation of confidence. For example if the target object is not present, this results in a very uniform pdf. In this case the confidence is very low and the target representation is not altered at all to circumvent overlearning. As a simple yet efficient implementation of the confidence measure, the absolute value of the pdf’s peak is utilized, which is approximated by the sample with the largest weight \( \pi^{(j)} \).

### 3 Smart Camera System

First we illustrate the Smart Camera’s architecture, afterwards we detail the benefits of this approach.

#### 3.1 Hardware Description

Our work is based on a mvBlueLYNX 420CX Smart Camera from Matrix Vision [10] which is shown in Fig. 3. The Smart Camera consists of a sensor, a FPGA, a processor and a networking interface. More precisely, it contains a single CCD sensor with VGA resolution (progressive scan, 12 MHz pixel clock) and an attached Bayer color mosaic. A Xilinx Spartan-IIIE FPGA (XC2S400E) is used for low-level processing. A 200 MHz Motorola MPC 8241 PowerPC processor with MMU & FPU running embedded Linux is used for the main computations. It further comprises 32 MB SDRAM (64 Bit, 100 MHz), 32 MB NOR-FLASH (4 MB Linux system files, approx. 40 MB compressed user filesystem) and 4 MB NAND-FLASH (bootloader, kernel, safeboot system, system configuration parameters). The Smart Camera communicates via a 100 MBit/s Ethernet connection, which is used both for field upgradeability and parameterization of the system and for transmission of the
tracking results during runtime. For direct connection to industrial controls 16 I/Os are available. XGA analog video output in conjunction with two serial ports are available, where monitor and mouse are connected for debugging and target initialization purposes. The form factor of the Smart Camera is (without lens) (w × h × l): 50 × 88 × 75 mm³. It consumes about 7 W power. The camera is not only intended for prototyping under laboratory conditions, it is also designed to meet the demands of harsh real world industrial environments.

3.2 Smart Camera Tracking Architecture

Fig. 4 illustrates the Smart Camera architecture and its output. In Fig. 5 the tracking architecture of the Smart Camera is depicted in more detail.

Figure 4. Smart Camera Architecture

Smart Camera’s Output

The Smart Camera’s output per iteration consists of:

- the mean state \( E[S_t] = \sum_{i=1}^{N} \pi_t^{(i)} X_t^{(i)} \), thus one value.
- the maximum likelihood state \( X_t^{(j)} \) with \( j | \pi_t^{(j)} = \max_{i=1}^{N} \{ \pi_t^{(i)} \} \) in conjunction with the confidence \( \pi_t^{(j)} \), resulting in two values.
- Optionally, a region of interest (ROI) around the sample with maximal likelihood can be transmitted too.

The whole output is transmitted via Ethernet using sockets. On the PC side, the data can be visualized on the fly and saved on hard disk for offline evaluation.

3.3 Benefits

This Smart Camera approach offers several benefits:

- **Low bandwidth requirements out of the camera:**
  The raw images are processed directly in the camera. Hence, only the approximated pdf of the target’s state has to be transmitted from the camera using relatively few parameters. This allows the use of standard networks (e.g., Ethernet) with virtually unlimited range. In our work, all the output summarizes to \( (N \star (k+1)+3) \) values per frame. For example, using \( N = 100 \) and constant velocity motion model \( (k=4) \), this leads to 503 values per frame. This is quite few data compared to transmitting all pixels of the raw image: For example (even un-demosaiced) VGA resolution needs about 307 k pixel values per frame. Even at (moderate) 15 fps this already leads to 37 MBit/s transmission rate, which is about 1/3 of the standard 100 MBit/s bandwidth.

- **No additional computing outside the camera has to be performed:**
  Each networking enabled external processing unit (a PC or a networking capable machine control in factory automation) does not have to deal with low level processing any more which algorithmically belongs to a camera. Instead it can concentrate on higher level processing using all Smart Cameras’ outputs as basis. Or such a unit can be used to passively supervise all outputs (e.g., in case of a PDA with WLAN in a surveillance application). Additionally, it becomes possible to connect the output of such a Smart Camera directly to a machine control unit (that does not offer dedicated computing resources for external devices), e.g., to a robot control unit for visual servoing. For this, the mean or the maximum likelihood state together with a measure for actual tracking confidence can be utilized directly for real-time machine control.
• **Higher resolution and framerate:** As the raw video stream does not need to comply with the camera’s output bandwidth any more, sensors with higher spatial or temporal resolutions can be used: due to the very close spatial proximity between sensor and processing means, higher bandwidth can be achieved more easily. In contrast, all scenarios with a conventional vision system (camera + PC) have their major drawbacks: First, transmitting the raw video stream in full spatial resolution at full frame rate to the external PC for doing the whole processing there can easily exceed today’s networking bandwidths. This applies all the more when multiple cameras come into play. Connections with higher bandwidths (e.g., CameraLink) on the other hand are too distance-limited (besides the fact that they are typically host-centralized). Second, if only regions-of-interest (ROIs) around samples induced by the particle filter were transmitted, the transmission between camera and PC would become part of the particle filter’s feedback loop. Indeterministic networking effects provoke that the particle filter’s prediction of samples’ states (i.e., ROIs) is not synchronous with the real world any more and thus measurements are done at wrong positions.

• **Multi-Camera systems:** As a consequence of above benefits, this approach offers optimal scaling for multi-camera systems to work together in a decentralized way. This enables large-scale camera networks e.g., for airport surveillance as they become reality today.

• **Small, Self-Contained Unit:** The Smart Camera approach offers a self-contained vision solution with a small form factor. This increases the reliability and enables the installation at size-limited places and on robot hands.

• **Parameterizability:** Our implementation allows for the parameterization of the particle filter in a wide spectrum. This comprises the number of samples $N$, the patch dimensions $(H_x,H_y)$, the number of histogram bins $B_{HS}, B_V$ (for $H \times S$ and for $V$), the blending factor $(HS \leftrightarrow V)$, the diffusion variance vector, the variance $\sigma^2$ for Bhattacharyya weighting and the motion model combination. Hence, an optimal (manual) adaption with respect to scene nature, real-time demands and available hardware resources can be found, even without recompiling the source files.

• **Adaptive Particle Filter’s Benefits:** A Kalman Filter implementation on a Smart Camera would also offer above benefits. However, it shows various drawbacks as it can only handle unimodal pdfs and linear models. As the particle filter approximates the – potentially arbitrarily shaped – pdf $p(X_t | Z_t)$ somewhat efficiently by samples, the bandwidth overhead is still moderate whereas the tracking robustness gain is immense. By adapting to slow appearance changes of the target with respect to the tracker’s confidence, the robustness is further increased.

4 **Results**

We will outline some results which are just an assortment of what is also available for download from the project’s website [11] in higher quality.

4.1 **Experimental Results**

For our first experiment, we initialize the camera with a cube object. It is trained by presenting it in front of the camera and saving the according color distribution as target reference. The tracking performance was convincing: Our Smart Camera is capable of robustly following the target over time at a framerate of over 15 fps at a sensor resolution of 640x480. For increased computational efficiency, the tracking directly runs on the raw and thus still Bayer color filtered pixels: Instead of first doing expensive Bayer demosaicing and finally only using the histogram which still contains no spatial information, we interpret each four-pixel Bayer neighborhood as one pixel representing RGB intensity (whereas the two green values are averaged), leading to QVGA resolution as tracking input. The final bandwidth demands are very moderate as the camera’s output consumes only about 15 kB/s (when using 100 samples). In the first experiment, a cube is tracked which is moved first vertically, then horizontally and afterwards in a circular way. The final pdf $p(X_t | Z_t)$ at time $t$ which the Smart Camera outputs is illustrated in Fig. 6, projected in x and y directions. Starting from this Figure, Fig. 7 illustrates several points in time in more detail: Concentrating on the circular motion part of this cube sequence, a screenshot of the samples’ actual positions in conjunction with their weights

![Figure 6. Experiment #1: pdf $p(X_t | Z_t)$ over iteration time $t$. Left: x-component, Right: y-component.](image-url)
is given. Note that we do not take advantage of the fact that the camera is mounted statically, i.e., no background segmentation is performed as a preprocessing step.

In the second experiment, we evaluate the performance of our Smart Camera in the context of surveillance: The Smart Camera is trained with a person’s face as target. It shows that the face can be tracked successfully in real-time too. Fig. 8 shows some results during the run.

The third experiment (Fig. 9) shows the performance of the Smart Camera tracking a person. Fig. 10 illustrates the coverage of the parameter space by the samples over time. Note how the background is covered by some samples (20% are chosen to be randomly distributed), whereas the path of the target is covered pretty strongly. Fig. 11 depicts the results of the person tracking sequence. Here, the adaption is parameterized very conservatively: Just the fact that a second person walks through the scene (iteration #94) lets the confidence drop so that target adaption is temporally prevented ($\alpha = 0$).

5 Conclusion

We presented a Smart Camera for embedded and adaptive object tracking in real-time. It is based on particle filtering on HSV color distributions and offers robust tracking performance as it can handle multiple hypotheses concurrently and adapt to slow appearance changes. Yet it offers a very bandwidth-conservative output, as only an approximation of the pdf $p(X_t | Z_t)$ is transmitted along with the mean and maximum likelihood state of the target. Thus, our tracking camera needs only about 15 kB/s bandwidth, which is less than 0.33% of the bandwidth than when the raw images were transmitted for external computation. Our output could be used directly, e.g., for connection to an industrial robot control unit, or for inter-camera communication on a higher level. Due to the low bandwidth requirements, it offers ubiquitous availability of the whole sensor network’s output, i.e., it is possible to acquire the output of all the cameras at any place in the network. The Smart Camera implementation is parameterizable in a wide spectrum to easily adapt to both hardware resources and scene...
properties. In the future, we plan to set up a multi-camera system to demonstrate also inter-camera communication on this higher level of abstraction (e.g., as basis for person forwarding in a surveillance application).

**Acknowledgment**

We would like to thank Sven Lanwer for his implementation work which is greatly appreciated. Also we would like to thank Matrix Vision for their generous support and the successful cooperation.

**References**


