A SMART CAMERA APPROACH TO REAL-TIME TRACKING

Sven Fleck, Sven Lanwer, Wolfgang Straßer

WSI/GRIS, University of Tübingen
Sand 14, 72076 Tübingen, Germany
phone: +(49) 7071 2970435, fax: + (49) 7071 295466, email: {fleck,lanwer,strasser}@gris.uni-tuebingen.de
web: www.gris.uni-tuebingen.de

ABSTRACT

Tracking applications using distributed sensor networks are emerging today, both in the field of surveillance (airports, 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 in real-time and offers a very bandwidth-conservative approach, as it only transmits the tracking results which are on a higher level of abstraction.

1. INTRODUCTION

Today's computer vision systems typically see cameras 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 physically embody 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 first prototype of a network-enabled Smart Camera capable of 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.

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 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:

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

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 $\mathbf{S}_t$. Each sample $\mathbf{s}$ consists of the state vector X and a weight , with $\sum_{i=1}^{N} {}^{(i)} = 1$. Thus, the i-th sample at time t is denoted by $\mathbf{s}_t^{(i)} = (X_t^{(i)}, {}_t^{(i)})$. Together they form the sample set $\mathbf{S}_t = \{\mathbf{s}_t^{(i)} | i = 1..N\}$.

Fig. 1 shows the principal operation of a Particle Filter with 8 particles, whereas its steps are outlined below.

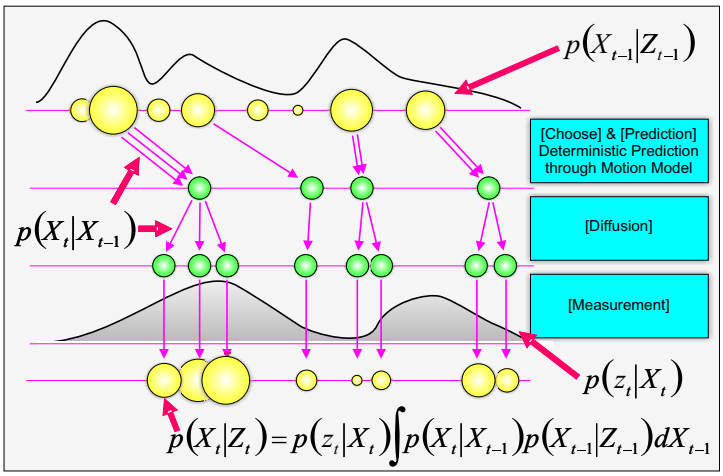

Figure 1: Particle Filter iteration loop

- **Choose Samples Step:** First, a cumulative histogram of all samples' weights is computed. Then, according to each particle's weight ${}_{t-1}^{(i)}$, 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:

$$p(X_t|Z_{t-1}) = \int p(X_t|X_{t-1})p(X_{t-1}|Z_{t-1})dX_{t-1} \qquad (2)$$

  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 ($X_t = AX_{t-1} + w_{t-1}$), each enriched with a Gaussian diffusion w_{t-1} to spread the samples and to allow for target moves differing from each motion model. Our state has the form $X_t^{(i)} = (x, y, v_x, v_y)_t^{(i)}$.
- **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 camera image).

$$p(X_t|Z_t) = p(z_t|X_t)p(X_t|Z_{t-1}) \qquad (3)$$

  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 (V) histogram 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 $\mathbf{s}_t^{(i)}$ induces an image patch $P_t^{(i)}$ 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 cases 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:

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

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

$$p_t^{(i)}[b] = Histo_{X_t^{(i)}}(b) = f \sum_{w\ P_t^{(i)}} k\left(\frac{\|w - \widetilde{X}_t^{(i)}\|}{a}\right) [I(w) - b]$$

with bin number b, pixel position w on the patch, bandwidth $a = \sqrt{H_x^2 + H_y^2}$ and normalization f, whereas $\widetilde{X}_t^{(i)}$ denotes the subset of $X_t^{(i)}$ which describes the (x,y) position in the image. The -function assures that each summand 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 made in histogram space.

From Histogram to new Weight

Now we compare the target histogram with each sample's histogram: For this, we use the popular Bhattacharyya similarity measure [4], both on the HS and the V histograms respectively:

$$[p_t^{(i)}[b], q_t[b]] = \sum_{b=1}^{B} \sqrt{p_t^{(i)}[b] q_t[b]}$$

with $p_t^{(i)}$ and q denoting the sample's and the target's histogram (respectively in HS and V space). Thus, the more similar a sample appears to the target, the larger becomes . These two similarities 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 and a 70 : 30 weighting between HS and V. Then, the Bhattacharyya distance

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

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

$$_t^{(i)} = \frac{1}{\sqrt{2}} \exp\left(-\frac{d_t^{(i)2}}{2\ ^2}\right) = \frac{1}{\sqrt{2}} \exp\left(-\frac{1 - [p_t^{(i)}[b], q[b]]}{2\ ^2}\right)$$

Hence, a low Bhattacharyya distance leads to a high probability weight $_t^{(i)}$ and thus the sample will be favored more in the next iteration.

3. SMART CAMERA SYSTEM

3.1 Hardware Description

We chose a mvBlueLYNX 420CX Smart Camera from Matrix Vision [10] as basis for our work which is shown in Fig. 2. The Smart

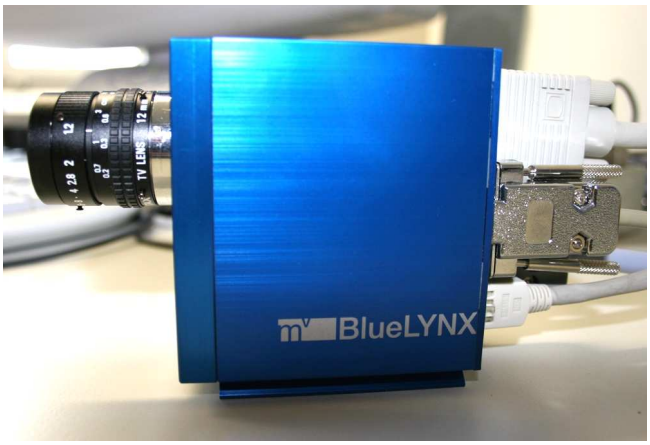

Figure 2: Our Smart Camera system.

Camera consists of a sensor, a FPGA, a processor and an Ethernet interface. More precisely, it contains a single CCD sensor with VGA resolution (progressive scan) and an attached Bayer color mosaic. A Xilinx Spartan-IIE FPGA is used for low-level processing. A 200 MHz Motorola PowerPC processor with MMU & FPU together with 32 MB SDRAM and 36 MB FLASH running Linux completes the system and communicates via a 100 MBit/s Ethernet connection for field upgradability and tracking result transmission. For direct connection to industrial controls, some I/Os are available. Analog video output in conjunction with two serial ports are available, where monitor and mouse are connected for debugging and target initialization purposes. 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.3 illustrates the Smart Camera architecture.

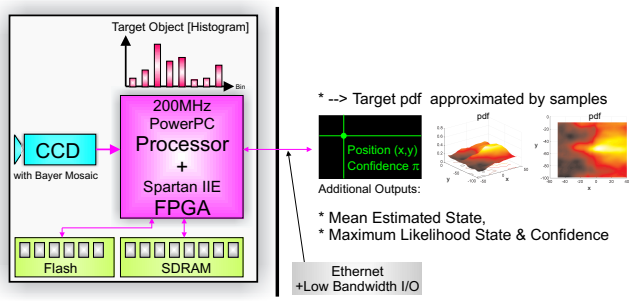

Figure 3: Smart Camera Architecture

Smart Camera's Output

The Smart Camera's output per iteration consists of:

- the pdf $p(X_t|Z_t)$, approximated by the sample set $\mathbf{S}_t = \{(X_t^{(i)}, \ _t^{(i)}), i = 1..N\}$. This leads to $(N*(k+1))$ values.

- the mean state $E[\mathbf{S_t}] = \sum_{\mathbf{i=1}}^{\mathbf{N}} \pi_{\mathbf{t}}^{(\mathbf{i})} \mathbf{X_t^{(i)}}$, 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.

Transmission

The Smart Camera's 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 image is 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 (i.e., Ethernet) with virtually unlimited range. In our work, all the output summarizes to $(N*(k+1)+3)$ values per frame. For example, using $N = 100$ and no velocity motion model ($k = 2$), this leads to 303 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 307k pixel values per frame. Even at (moderate) 15 fps this already leads to 37 MBit/s transmission rate, which is about $1/3$ of 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 algorithms 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 likely 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 bandwidth (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 in 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 (in H, S,V), the blending factor (HS+V), the diffusion variance vector, the variance for Bhattacharyya weighting and the motion model combination.
- **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.

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 15fps at 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.4, projected in x and y directions. Starting from this Figure,

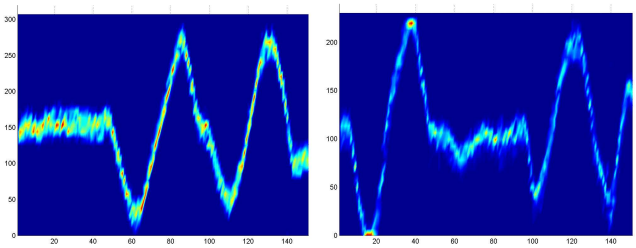

Figure 4: pdf $p(X_t|Z_t)$ over iteration time t. **Left:** x-component, **Right:** y-component.

Fig.5 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 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 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.6 shows some results during the run.

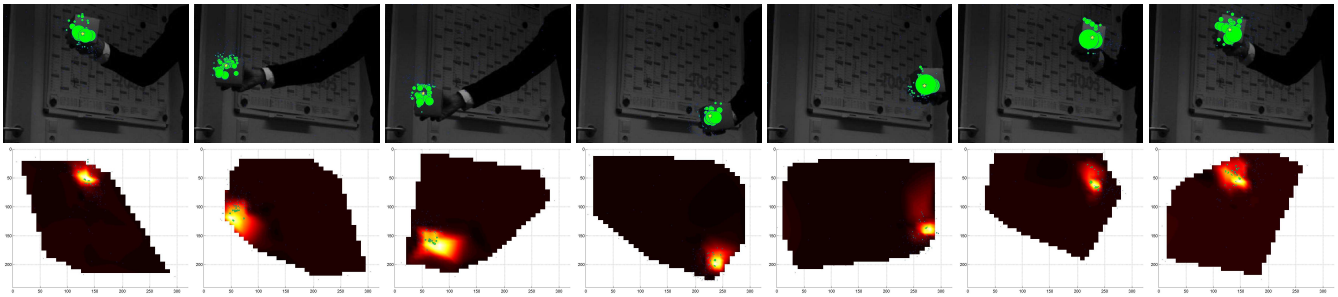

Figure 5: Circular Motion sequence of experiment #1. Image (**Top row**) and approximated pdf (**Bottom row**) at iteration #100, 109, 113, 125, 129, 136, 141. Samples are shown in green, the mean state is denoted as yellow star.

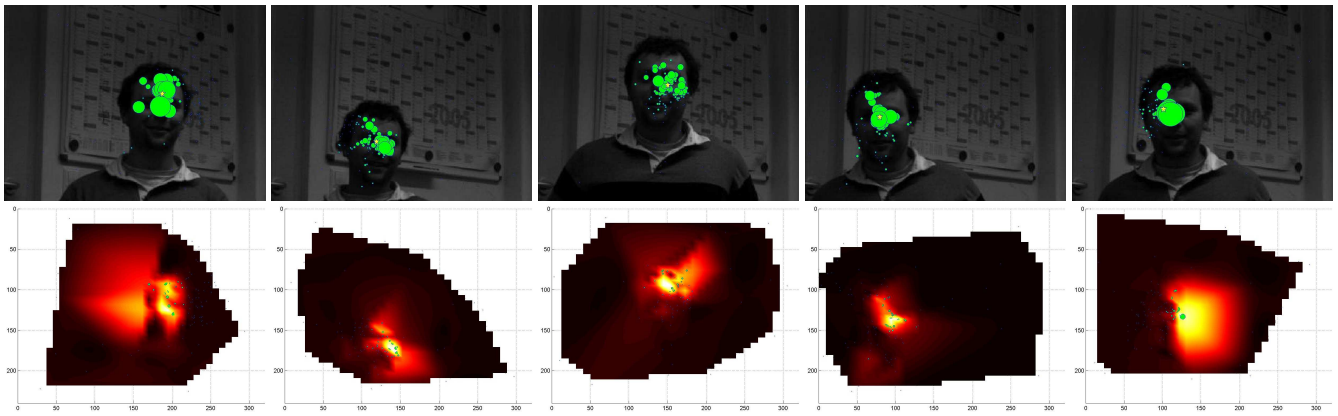

Figure 6: Experiment #2: Face tracking sequence. Image (**Top row**) and approximated pdf (**Bottom row**) at iteration #18, 35, 49, 58, 79.

5. CONCLUSION

We presented a Smart Camera for real-time object tracking. Using Particle Filtering on HSV color distributions it offers robust tracking performance as it can handle multiple hypotheses concurrently. Yet it offers a very bandwidth-conservative output, as only the approximated pdf is transmitted along with the mean and maximum likely state of the target. Thus, our tracking camera needs only about 15kB/s bandwidth. This prevents the expensive transmission of raw video streams as basis for external computation. Our output could be directly used, 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. First extension will be to extend the particle state to include scale changes. Additionally, future work includes the automatic adaption of target appearance during runtime to increase the tracking robustness with respect to illumination changes. Furthermore, 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 Matrix Vision for their generous support and successful cooperation.

REFERENCES

[1] N. D. F. Arnaud Doucet and N. Gordon, *Sequential Monte Carlo Methods in Practice*. Springer Verlag, 2001.

[2] "Special issue on: Sequential state estimation: From Kalman filters to particle filters," *Proceedings of the IEEE*, vol. 92, no. 3, 2004.

[3] M. Isard and A. Blake, "Condensation – conditional density propagation for visual tracking," *International Journal of Computer Vision*, vol. 29, no. 1, pp. 5–28, 1998.

[4] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-based object tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 05, pp. 564–575, 2003.

[5] K. Okuma, A. Taleghani, N. de Freitas, J. J. Little, and D. G. Lowe, "A boosted particle filter: Multitarget detection and tracking," in *ECCV 2004: 8th European Conference on Computer Vision*, 2004.

[6] K. Nummiaro, E. Koller-Meier, and L. V. Gool, "A color based particle filter," in *First Int. Workshop on Generative-Model-Based Vision GMBV'02*, 2002.

[7] P. Prez, C. Hue, J. Vermaak, and M. Gangnet, "Color-based probabilistic tracking," in *European Conference on Computer Vision, ECCV'2002, LNCS 2350*.

[8] P. Prez, J. Vermaak, and A. Blake, "Data fusion for visual tracking with particles," *Proceedings of IEEE (Issue on State Estimation)*, vol. 92, no. 3, pp. 495–513, 2004.

[9] M. Spengler and B. Schiele, "Towards robust multi-cue integration for visual tracking," *Lecture Notes in Computer Science*, vol. 2095, p. 93ff., 2001.

[10] "Matrix vision," http://www.matrix-vision.com.

[11] "Project's website," www.gris.uni-tuebingen.de/~sfleck/matrixtracking.