Real–Time Pedestrian Tracking in Natural Scenes

Joachim Denzler and Heinrich Niemann

denzler,niemann@informatik.uni-erlangen.de

The following paper was published in the
Proceedings on the 7^{th} International Conference
on
Computer Analysis of Images and Patterns

Kiel, September $10^{th} - 12^{th}$, 1997

Real–Time Pedestrian Tracking in Natural Scenes

J. Denzler, H. Niemann

Universität Erlangen–Nürnberg
Lehrstuhl für Mustererkennung (Informatik 5)
Martensstr. 3, D–91058 Erlangen

Abstract In computer vision real–time tracking of moving objects in natural scenes has become more and more important. In this paper we describe a complete system for data driven tracking of moving objects. We apply the system to tracking pedestrians in natural scenes. No specialized hardware is used. To achieve the necessary efficiency several principles of active vision, namely selection in space, time, and resolution are implemented. For object tracking, a contour based approach is used which allows contour extraction and tracking within the image frame rate on general purpose architectures. A pan/tilt camera is steered by a camera control module to pursue the moving object. A dedicated attention module is responsible for the robustness of the complete system. The experiments over several hours prove the robustness and accuracy of the whole system. Tracking of pedestrians in a natural scene has been successful in 79% of the time.

1 Introduction

The development of machines which interact with their environment, has become more and more important in the past years. One important aspect of those machines, especially if they are able to move autonomously, is the capability to detect and to track moving objects, even if the system is moving itself, too.

There exist some sophisticated systems for tracking moving objects in real–time. [11] present a stereo camera system, which pursues moving objects in indoor natural scenes. The approach of [2] realizes a contour tracker, which pursues a moving toy car in a laboratory environment with cooperative background. Tracking cars on highways in real–time has been the topic of [10, 9]. An optical flow based realization of real–time tracking can be found in [3].

All these system have in common, that they use specialized hardware like pipelined image processors or transputer networks. Only a few systems are known for real–time tracking on general purpose architectures, for example [1, 6]. In this paper we describe a complete system for data driven detection and tracking of moving objects in real–time on a general purpose processor. Principles of active vision are included in different modules of the system, in order to cope with real–time constraints. No specialized hardware is used. We demonstrate the quality of the system for tracking moving pedestrians in outdoor scenes. Due to the nature of outdoor scenes the system has to cope with changing illuminations, reflections and other moving objects.

The paper is structured as follows. In Sect. 2 we give an overview over the system and describe the two stages in which the system operates. Sect. 3 describes in more detail the tracking algorithm. Sect. 4 demonstrates the quality of the system for natural

scenes, even during sunshine, rain and snow fall. The papers concludes in Sect. 5 with a discussion of the approach and an outlook to future work.

2 System Overview

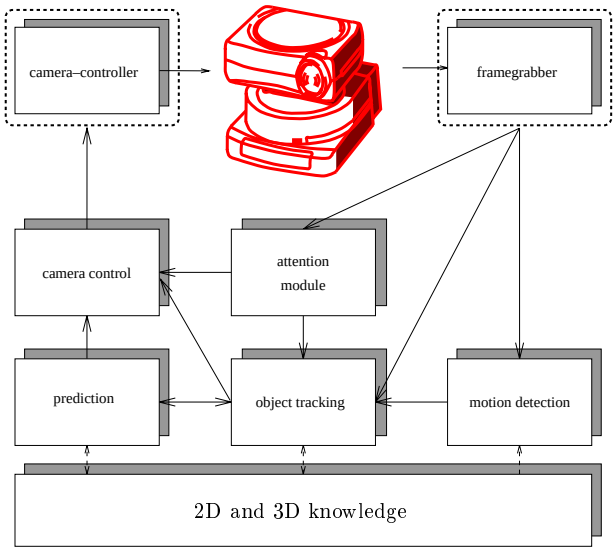

Figure1. Overview of the complete system

In Figure 1 an overview of the complete system for data driven tracking is shown. Two hardware dependent modules can be seen, the frame-grabber and the physically control of the camera. The other modules, the motion detection module, the object tracking module, the prediction module, the camera control module, and the attention module are implemented in C++ and use general purpose processors. In addition, a 2D– and 3D knowledge base is available, where problem dependent knowledge can be introduced into the system. This kind of knowledge is not used for the application presented in this paper.

The system runs in two stages: an *initialization stage*, where the motion detection modules detects motion in the scene, and a *tracking stage*. During the initialization stage, the camera is static to allow for a simple difference image operation for motion detection in the scene. For the 128×128 low resolution difference image a binary image is computed. This is smoothed to get a few significant regions in the image, where changes have occurred. The largest of these regions in size is selected and used as initialization for the tracking stage.

During the tracking stage all modules work together and communicate with each other. This is indicated by the connections between the modules. The object tracking

module (a detailed description follows in Sect. 3) tracks the contour of the moving object, and uses predictions computed by the prediction module. Only a small window of the camera image containing the moving object is processed. The center of gravity of the object's contour is sent to the camera control module to steer the camera. The motion detection module computes with a low frame rate independently moving objects, using knowledge about the camera motion [8]. The robustness of the whole system is realized by the attention module. This module watches over the system to detect errors during tracking. For this, in certain intervals features are computed for the contour of the moving object (x– and y–moments of the contour). Rapid changes in the moments give hints for errors in the contour extraction.

As soon as an error is detected, the attention module stops tracking, sends a signal to the camera control module to stop camera motion, and then switches back to the initialization stage.

3 Object Tracking

For object tracking we have developed a new algorithm, called *active rays*. This algorithm is motivated by active contours and is based on the principle of contour extraction and tracking. In contrast to [9], who uses also a radial representation of the contour, we define an internal energy similar to the approach of active contours, which couples neighboring contour points.

For active rays, we first have to define a reference point $\boldsymbol{m} = (x_m, y_m)^T$, which has to lie inside the image contour. An active ray $\varrho_{\boldsymbol{m}}(\phi, \lambda)$ is defined on the image plane (x, y) as a 1D function depending on those gray values $f(x, y)$ of the image, which are on a straight line from the image point $\boldsymbol{m}$ in direction ϕ

$$\varrho_{\boldsymbol{m}}(\phi, \lambda) = f(x_m + \lambda \cos(\phi), y_m + \lambda \sin(\phi)), \tag{1}$$

with $0 \leq \lambda \leq n_\phi$, where n_ϕ is given by the diagonal of the image. Now, by assuming,

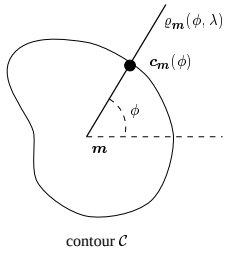

Figure2. Representation of a contour point by active rays

that each ray only hits the object contour once, we can identify a point of the contour

by the parameter $\lambda^*(\phi) \geq 0$

$$\begin{aligned}\lambda^*(\phi) &= \underset{\lambda}{\operatorname{argmin}}\Big(-|\nabla f(x_m + \lambda\cos(\phi), y_m + \lambda\sin(\phi))|^2\Big) \\ &= \underset{\lambda}{\operatorname{argmin}} \left(-\left|\frac{\partial}{\partial\lambda}\varrho_{\boldsymbol{m}}(\phi,\lambda)\right|^2\right), \end{aligned} \tag{2}$$

with $0 \leq \phi < 2\pi$. If the ray hits the contour many times (for example for concave contours) we have to introduce a set of values $\lambda^*(\phi)$. This is not important for the application described here. The representation by active rays is illustrated in Figure 2. The step, which lead to (2), is motivated by the assumption, that an edge in 2D can also be found by a gradient search in the corresponding 1D signal. Of course, edges which are in the direction ϕ from the reference point cannot be found on the ray $\varrho_{\boldsymbol{m}}(\phi,\lambda)$. The experiments will show, that this case is not relevant in practice. Having the optimal value for $\lambda^*(\phi)$ the contour point $\boldsymbol{c_m}(\phi)$ in the image plane can easily be computed by

$$\boldsymbol{c_m}(\phi) = (x_m + \lambda^*(\phi)\cos(\phi), y_m + \lambda^*(\phi)\sin(\phi)), \tag{3}$$

with $0 \leq \phi < 2\pi$. The most important aspect of this approach, especially for real–time applications, is the reduction of the contour point search from the 2D image plane to a 1D signal. This reduces the computation time which will be shown in the experimental part of this paper.

In Figure 3, left the extracted contour of an object and in Figure 3, right the function $\lambda^*(\phi)$ are shown. One can observe, that the function $\lambda^*(\phi)$ is smooth for the angles which corresponds to the correctly extracted contour ($0 - 4/3\pi$). Then, an error can be seen, both in the extracted contour and in the function $\lambda^*(\phi)$. For $\phi \in [4/3\pi, 3/2\pi[$ the function is not smooth, because a wrong contour has been extracted. This is no surprise. Looking at equation (2) one can see, that up to now, the contour points are calculated without taking into account neighboring contour elements. Thus, we need to introduce some linkage between neighboring contour points to take into consideration that normally contours are coherent in space, i.e. that contours are smooth. A usual approach to connect neighboring contour points together is to introduce an internal energy similar to the active contour approach.

An internal energy which handles the above mentioned demands is

$$E_i(\boldsymbol{c_m}(\phi)) = \frac{\alpha(\phi)\left|\frac{d}{d\phi}\lambda(\phi)\right|^2 + \beta(\phi)\left|\frac{d^2}{d\phi^2}\lambda(\phi)\right|^2}{2}. \tag{4}$$

This energy also depends only on a 1D function, compared to active contours [7], where the corresponding energy depends on a 2D one. This energy can be formally derived from the internal energy definition of active contours. This is beyond the scope of this paper and is published elsewhere [5]. Now we have an energy, which describes contour point candidates for each ray and an energy, which connects the rays to get a smooth contour. Similar to active contours we define a total energy E

$$E = \int_0^{2\pi} \left[\frac{\alpha(\phi)\left|\frac{d}{d\phi}\lambda(\phi)\right|^2 + \beta(\phi)\left|\frac{d^2}{d\phi^2}\lambda(\phi)\right|^2}{2} - \left|\frac{d}{d\lambda}\varrho_{\boldsymbol{m}}(\phi,\lambda)\right|^2\right] d\phi. \tag{5}$$

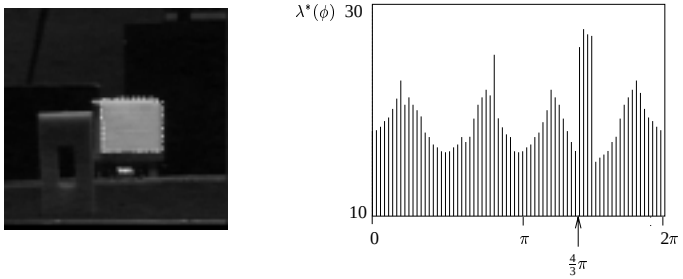

Figure3. Left: 2D contour extracted by active rays. Right: 1D function λ^* of the corresponding 2D contour shown on the left.

The contour extraction can then be described as an energy minimization problem. Using the variational approach the Euler–Lagrange differential equation

$$\alpha(\phi)\frac{d^2}{d\phi^2}\lambda(\phi) - \beta(\phi)\frac{d^4}{d\phi^4}\lambda(\phi) + \frac{d}{d\lambda}\left|\frac{d}{d\lambda}\varrho_{\boldsymbol{m}}(\phi,\lambda)\right|^2 = 0$$

must be solved. Again, this differential equation depends on a 1D function, in contrast to the same differential equation for active contours.

Some remarks must be done regarding the reference point $\boldsymbol{m}$. As already mentioned, this point must lie inside the object contour, but the position may be arbitrary. For a prediction step, a unique position would be of great advantage. Thus, we always choose the center of gravity of the contour

$$\boldsymbol{m} = 1/2\pi \int_0^{2\pi} \boldsymbol{c_m}(\phi)\, d\phi \tag{6}$$

for the reference point. If this equation does not hold for an actual reference point and an extracted contour, we update the reference point using equation (6), and restart the contour extraction with the updated reference point's position.

But how can we compute the reference point for the first image, if we do not know the contour of the object? This is no problem, because we apply this algorithm to object tracking. Assuming a static camera during motion detection, we always have a coarse idea, where a moving object is located in the image. The approach we have chosen is a difference image algorithm (see the motion detection module in the previous Sect.), which computes areas in the image, where changes occur. The center of such an area is taken a initial reference point $\boldsymbol{m}$ for the first image.

Up to now we have shortly summarized the theory of active rays for the continuous case. By applying active rays for contour extraction in images we have to go to the discrete case. Two approaches for the discretization are possible: fixed sampling rate $\triangle\phi$ for the angle ϕ or the so called any–time behavior. Due to lack of space the reader is referred to [4]

4 Experimental Environment and Results

We have chosen an outdoor scene to evaluate the performance of the complete system as well as the robustness and quality of the tracking algorithm by active rays.

The system runs on a SGI Onyx with two R10000 processors without any specialized hardware. The camera is a pan/tilt device Canon VCC1 which is connected with the workstation via RS232. A similar tracking performance can be achieved with a HP 735/99 MHz, but this architectures does not allow for full frame rate image grabbing.

We have manually evaluated 40 minutes of tracking. Every 20th frame has been written to disk after the experiments, i.e. 3000 images have been evaluated. We judged visually, whether or not the active ray has correctly extracted the moving object. There have been 73 automatic initializations in which 34 initializations fail. The reason is, that the center of gravity of the binary motion region not always lie inside the moving object. For example, for two objects moving close together, the binary region covers both object and the center of gravity lies between the objects. This has been expected, because we apply a very fast and simple algorithm for motion detection.

After correct initialization (39 times), we could correctly track 20 moving persons for 8680 images as long as they have been visible (average number of images: 434, i.e. 17 seconds). Caused by an error in the tracking algorithm, in 19 cases the object was lost after an average number of 265 images, i.e. after 10 seconds which results in another 5040 correctly tracked images. Most of the errors occur near the end of the place, because there are a lot of strong background edges (the trees).

After loosing the object, the attention module needs an average number of 91 images (i.e. 3.6 sec), for detecting the loss of the object; in 26 of the 39 cases the detection time was below one second. Hence, in 3560 images the system has not tracked a moving person, although it has been in the tracking stage. Thus, we get a total success rate for tracking of 79 %.

In Figure 4 some typical result can be seen. In the last row, the moving person moves outside the field of view which the camera can cover by pan/tilt movements. In Figure 5 some additional results can be seen. In the first row the effect of the data driven tracking is illustrated. The system tracks the moving contour of the small snow-plow, because it cannot distinguish between contours of moving pedestrians and some other contours. For this, knowledge is necessary which can be added into the 2D and 3D knowledge base (see Sect. 2). Tracking one person out of the crowd also works, if the initialization is correct (Figure 5, second row).

5 Conclusion and Future Work

In this paper we have presented a complete system for data driven object tracking. Several principles of active vision are implemented, which allows — in contrast to [9] — for real–time tracking without specialized hardware. The important aspect of this system are the tracking module, which uses a new algorithm for contour extraction and tracking working within the image frame rate, and the attention module, which watches over the complete system to detect errors. Thus, a very robust tracking over a long time can be performed. The system satisfies two key components, stated by [11], namely

Figure4. Results for tracking moving pedestrians

- the continuous operation over time and real–time response to different types of events and
- an open and expandable design due the object–oriented implementation.

The emphasis on real–time performance on general purpose hardware has led us to consider simple algorithms, especially for the motion detection and motion tracking module. However, we argue, that the combination of simple methods and an attention module, which detects errors, can lead to a robust performance. Our experiments on natural scenes support this claim.

The system can be applied to gate control problems. The tracking algorithm is robust against changes in the camera parameters, especially against zooming during tracking. This means, that during the tracking a synchronous zooming toward the moving object can be done, for example to extract features of the face to identify the person. This will be our short time goal. In addition, we are working on integrating task specific knowledge into the knowledge base of the system, in order to increase the contour extraction and tracking result. One idea is to limitate the normalized shape of the contour, i.e. the function $\lambda^*(\phi)$, to certain values, which corresponds to typical contours of moving pedestrians.

Figure5. Results for tracking moving pedestrians

References

1. M. Armstrong and A. Zisserman. Robust object tracking. In *Second Asian Conference on Computer Vision*, pages I/58–I/62, Singapore, 1995.
2. R. Curwen and A. Blake. Dynamic contours: Real–time active splines. In A. Blake and A. Yuille, editors, *Active Vision*, pages 39–58. MIT Press, Cambridge, Massachusetts, London, England, 1992.
3. K. Daniilidis, M. Hansen, C. Krauss, and G. Sommer. Auf dem Weg zum künstlichen aktiven Sehen: Modellfreie Bewegungsverfolgung durch Kameranachführung. In *DAGM 1995, Bielefeld*, pages 277–284, 1995.
4. J. Denzler. *Active Vision for Real–Time Object Tracking*. Dissertation, Technische Fakultät, Universität Erlangen–Nürnberg, Erlangen, June, 1997.
5. J. Denzler and H. Niemann. Echtzeitobjektverfolgung mit aktiven Strahlen. In *Mustererkennung, 1996*, pages 84–91, Heidelberg, 1996.
6. G.D. Hager and K. Toyama. X vision: Combining image warping and geometric constraints for fast visual tracking. In A. Blake, editor, *Computer Vision - ECCV 96*, pages 507–517, Berlin, Heidelberg, New York, London, 1996. Lecture Notes in Computer Science.
7. M. Kass, A. Wittkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 2(3):321–331, 1988.
8. D. Murray and A. Basu. Motion tracking with an active camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):449–459, 1994.
9. S.M. Smith and J.M. Brady. Asset-2: Real–time motion segmentation and shape tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):814–820, 1995.
10. F. Thomanek and E.D. Dickmanns. Autonomous road vehicle guidance in normal traffic. In *Second Asian Conference on Computer Vision*, pages III/11–III/15, Singapore, 1995.
11. T. Uhlin, P. Nordlund, A. Maki, and J.O. Eklundh. Towards an active visual observer. In *International Conference on Computer Vision*, pages 679–686, Cambridge, Massachusetts, 1995.

This article was processed using the LaTeX macro package with LLNCS style