

Towards High-Speed Vision for Attention and Navigation of Autonomous City Explorer (ACE)

Tingting Xu, Tianguang Zhang, Kolja Kühnlenz and Martin Buss
*Institute of Automatic Control Engineering
Technische Universität München
80333, Munich,
Germany*

1. Introduction

In the project Autonomous City Explorer (ACE) a mobile robot should autonomously, efficiently and safely navigate in unstructured urban environments. From the biological aspect, the robot should not only plan its visual attention to acquire essential information about the unknown real world but also estimate the ego motion for the navigation based on vision with definitely fulfilled real-time capability. To achieve this, a multi-camera system is developed, which contains a multi-focal multi-camera platform, the camera head, for attentional gaze control and two high-speed cameras mounted towards the grounds for accurate visual odometry in extreme terrain.

How to apply the human visual attention selection model on a mobile robot has become an intensively investigated research field. An active vision system should autonomously plan the robot's view direction not only based on a specific task but also for stimulus-based exploration of unknown real-world environment to collect more information. Moreover, psychological experiments also show that the familiarity of the current context also strongly influences the human attention selection behavior. To solve this context-based attention selection problem, we propose a view direction planning strategy based on the information theory. This strategy combines top-down attention selection in 3D space and bottom-up attention selection on the basis of a 2D saliency map. In both spaces the information content increases are defined. The optimal view direction is chosen which results in a maximum information gain after a camera view direction change. The main contribution is that a concerted information-based scalar is inserted to evaluate the information gains in the both sides. Moreover, the robot behavior, the choice of attention selection mechanism, can be adaptive to the current context.

In addition, we implemented the compute-intensive bottom-up attention on Graphics Processing Units (GPUs) using the Compute Unified Device Architecture (CUDA), which provides an excellent speed-up of the system, due to the highly parallelizable structure. Using 4 NVIDIA GeForce 8800 (GTX) graphics cards for the input images at a resolution of 640 x 480, the computational cost is only 3.1ms with a frame rate of 313 fps. The saliency map generation on GPUs is approximately 8.5 times faster than the standard CPU implementations.

Angle-encoders on the wheels of the platform are normally used for the odometry. But if ACE moves on the ground which is not flat or has sands, it will slide. The encoders can not provide accurate information any more. Using a high-speed camera with 200 Hz, an elaborated concept for visual odometry based on optical flow is implemented. Utilizing Kalman Filter for data fusion, a distinctly local, low-latency approach that facilitates closed-loop motion control and highly accurate dead reckoning is proposed. It helps ACE determine the relatively precise position and orientation. Image processing at high frequency can decrease the time delay of close-loop control and improve the system stability. The various vision-based modules enable an autonomous view direction planning as well as visual odometry in real time. The performance is experimentally evaluated.

2. Overview of ACE and its high-speed vision system

The ACE project (Lidoris et al., 2007) (see Fig. 2.1 left) envisions to develop a robot that will autonomously navigate in an unstructured urban environment and find its way through interaction with humans. This project combines the research fields of robot localization, navigation, human-robot interaction etc..

Seen from the biological aspect, the visual information provided by the camera system on ACE is very essential for attention as well as navigation. Another prerequisite of the vision system is the image processing efficiency. The real-time requirement should be fulfilled during the robot locomotion.

The vision system of ACE consists of a multi-focal stereo camera platform (Fig. 2.2 middle) for the interaction and attention and a high-speed camera (Fig. 2.2 right) for visual odometry. The camera platform comprises several vision sensors with independent motion control which strongly differ in fields of view and measurement accuracy. High-speed gaze shift capabilities and novel intelligent multi-focal gaze coordination concepts provide fast and optimal situational attention changes of the individual sensors. Thereby, large and complex dynamically changing environments are perceived flexibly and efficiently. The detailed description of the camera platform is in (Kühnlenz, 2006a; Kühnlenz, 2006b). Currently in our application, only the wide-angle stereo-camera is used to demonstrate the attentional saccade caused by the saliency in the environment.

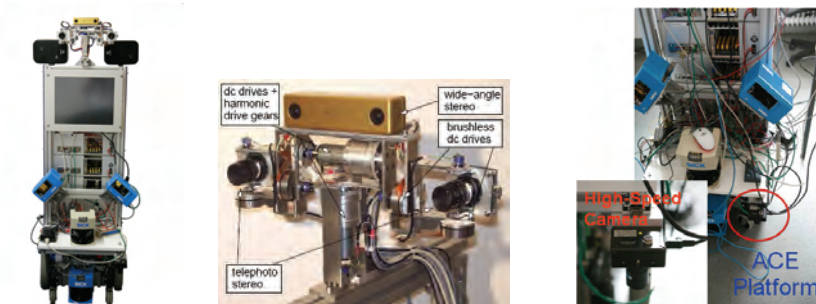


Fig. 2.1. Autonomous City Explorer (ACE) (left), the camera platform (middle) and the high-speed camera (right)

Moreover, a dragonfly® express camera (Point Grey Research Inc.), fitted with a normal wide-angle lens, is mounted on the mobile platform, facing the ground. This camera can work at 200 fps with the resolution of 640x480 pixels and be applied for visual odometry.

3. Information-based visual attention of mobile robots

3.1 Related work

In the robotics domain a variety of approaches to the view direction planning of active vision systems has been already proposed. The most concepts are based on the predefined robot tasks and in a top-down way. Above all, robot self-localization using active vision is well studied. In (Davison, 1998) visual information is used for simultaneous localization and map-building for a robot operating in an unknown environment. Point features are used as visual landmarks. The active cameras can re-detect the previously seen features and adjust their maps. In (Pellkofer & Dickmanns, 2000) an approach to an optimal gaze control system for autonomous vehicles is proposed in which the perceptive situation and subjective situation besides the physical situation are also predicted and the viewing behavior is planned and optimized in advance. For gaze control of humanoid robot the basic idea of (Seara & Schmidt, 2005) is based on maximization of the predicted visual information content of a view situation. A task decision strategy is applied to the view direction selection for individual tasks.

In the last few years, bottom-up saliency based attention selection models also become focus of robot view direction planning. A saliency map model was firstly proposed in (Itti et al., 1998). In the saliency map model the salient positions in a static image are selected by low-level features. The saliency map predicts the bottom-up based visual attention allocation. No high-level object recognition is required to drive a robot's attention, if bottom-up signals are also taken into account.

By now, the top-down and the bottom-up attention selections are only combined in the 2D image-space. In (Ouerhani et al., 2005) a visual attention-based approach is proposed for robot navigation. The trajectory lengths of the salient scene locations are regarded as a criterion for a good environment landmark. In (Frintrop, 2006) a biologically motivated computational attention system VOCUS is introduced, which has two operation modes: the exploration mode based on strong contrasts and uniqueness of a feature and the search mode using previously learned information of a target object to bias the saliency computations with respect to the target. However, the task accomplishment is evaluated in the image space which can only contain the information which is currently located in the field of view, although the performance evaluation in robotics domain is usually executed in the task-space.

Another key factor which has an influence on attention mechanism is the scene context. The context has already been used to facilitate object detection in the natural scenes by directing attention or eyes to diagnostic regions (Torralba & Sinha, 2001) and scene recognition (Im & Cho, 2006). In both cases the scene context is only statically observed. In (Remazeilles & Chaumette, 2006) vision-based navigation using environment representation is proposed. An image memory, a database of images acquired during a learning phase, is used to describe the path which the robot should follow. However, there is no dynamical context-based behavior adaptation considered.

3.2 Strategy overview

The objective is to plan the robot view direction with visual information from the input image, considering the competition of the task-based top-down attention and the stimulus-based bottom-up attention as well as behavior adaptation on the current context. Fig. 3.1 illustrates the view direction planning strategy architecture.

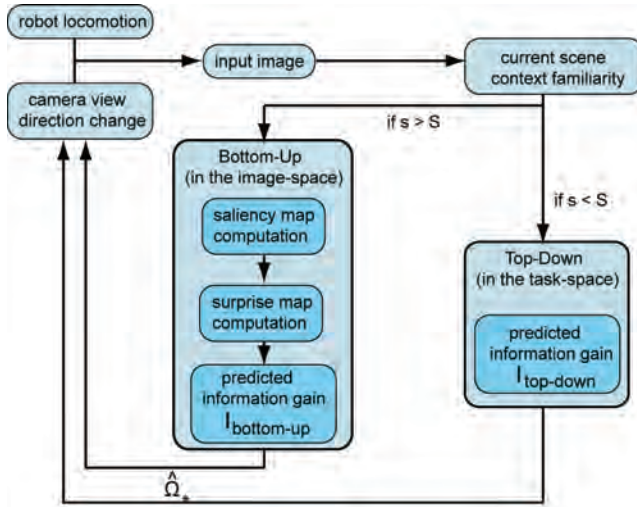


Fig. 3.1. View direction planning strategy architecture

We define the optimal view direction as the view direction with the estimated maximum information gain, calculated as the relative entropy/Kullback-Leibler (KL) divergence.

$$\hat{\Omega}_*^{k+1|k} = \arg \max_{\hat{\Omega}} (v(s) \cdot I_{top-down}(\hat{\Omega}) + (1 - v(s)) \cdot I_{bottom-up}(\hat{\Omega})) \quad (1)$$

with

$$v(s) = \begin{cases} 1 & \text{if } s < S \\ 0 & \text{if } s > S \end{cases} \quad (2)$$

$I_{top-down}$ and $I_{bottom-up}$ indicate the relative entropies acquired from the top-down and bottom-up sides with $v(s)$ the context-based weighting factor for the top-down attention. The total visual attention is 100%. Therefore, the weight of the bottom-up attention is $1 - v(s)$. The detailed definition of $v(s)$ is described in Section 3.5. $\hat{\Omega}$ and $\hat{\Omega}_*$ stand for the possible view directions and the optimal view direction of the camera.

3.3 Information-based modeling of the top-down attention

For the top-down attention we model the system state \underline{x} as 2D Gaussian distributions with the average value $\underline{\mu}$ and the covariance matrix $R_{\underline{x}}$ in the task-space. p and q are the prior and the predicted posterior probability density functions (pdf) with the continuous variable \underline{x} for specific tasks

$$p(\underline{x}) = \frac{1}{(\sqrt{2\pi})^n \cdot |R_{\underline{x}}^k|} \cdot \exp\left(-\frac{1}{2}(\underline{x}^k - \underline{\mu}^k)^T \cdot R_{\underline{x}}^{k-1}(\underline{x}^k - \underline{\mu}^k)\right) \quad (3)$$

and

$$q(\underline{x}, \hat{\Omega}) = \frac{1}{(2\pi)^n \cdot |R|} \cdot \exp\left(-\frac{1}{2}(\underline{x}^{k+1/k} - \underline{\mu}(\hat{\Omega})^{k+1/k})^T R^{-1} \cdot (\underline{x}^{k+1/k} - \underline{\mu}(\hat{\Omega})^{k+1/k})\right) \quad (4)$$

with the dimension n of the state variable \underline{x} and with

$$R = R_{\underline{x}}(\hat{\Omega})^{k+1/k} . \quad (5)$$

The relative entropy is then computed as follows:

$$I_{top-down} = KL(p_{top-down} \mid \mid q_{top-down}) = \iint_{\mathcal{D}} p(\underline{x}) \cdot \log \frac{p(\underline{x})}{q(\underline{x}, \hat{\Omega})} \text{ in [bit]} \quad (6)$$

3.4 Information-based modeling of the bottom-up attention

Besides the task accomplishment, the robot should also have the ability to explore the world, acquire more information, update the knowledge and also react to the unexpected events in the environment. In order to achieve this, a bottom-up attention selection is integrated. Here we consider the static outliers as well as the temporal novelty in the image-space.

For the static outliers we use the saliency map model proposed in (Itti et al., 1998). As known, human is much more attracted by salient objects than by their neighbourhood. The bottom-up saliency map is biology-inspired and can predict the position of the salient regions in a real-scene image.

In Fig. 3.2 the saliency map model is visualized. Firstly, an input image is sub-sampled into a dyadic Gaussian pyramid in three channels (intensity, orientation for 0° , 45° , 90° , 135° , opponent colour in red/green and blue/yellow). Then a centre-surround difference is calculated for the images in the Gaussian pyramid. In this phase feature maps are generated in which the salient pixels with respect to their neighbourhood are highlighted. Using across-scale combinations the feature maps are combined and normalized into a conspicuity map in each channel. The saliency map is the linear combination of the conspicuity maps. The bright pixels are the salient and interesting pixels predicted by the saliency map model. For the temporal novelty we applied a similar Bayesian definition like (Itti & Baldi, 2005) for the information content of an image, but directly on the saliency map. The notion "surprise" is used here to indicate the unexpected events. Only the positions spatially salient and temporally surprising are taken to draw the robot's attention. Therefore, we build a surprise map on two consecutive saliency maps without camera movement to find the unexpected event.

Firstly, as an example, the saliency maps of images at the resolution of 640×480 are rescaled into 40×30 pixels. Thus, each pixel represents the local saliency value of a 16×16 region.

Secondly, we model the data D received from the saliency map as Poisson distribution $M(\lambda(x_i, y_i))$. $\lambda(x_i, y_i)$ stands for the saliency value with $x_i = 1, \dots, 40$ and $y_i = 1, \dots, 30$. Therefore, a prior probability distribution $p_i(x_i, y_i)$ can be defined as a Gamma probability density (Itti & Baldi, 2005) for the i -th pixel:

$$p_i(x_i, y_i) = \gamma(\lambda, \alpha, \beta) = \frac{\beta^\alpha \lambda^{\alpha-1} e^{-\beta\lambda}}{\Gamma(\alpha)} \quad (7)$$

with the shape $\alpha > 0$, the inverse scale $\beta > 0$, and $\Gamma(\cdot)$ the Euler Gamma function.

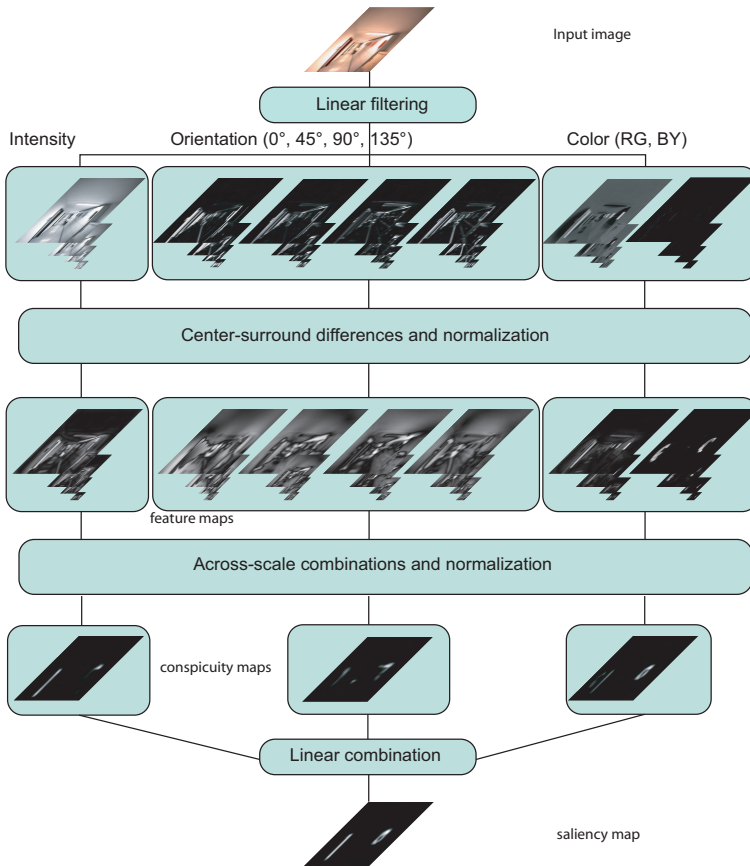


Fig. 3.2. The saliency map computation model

The posterior probability distribution $p((x_i, y_i) | D)$ is obtained from the 2. saliency map with the new saliency value $\lambda'(x_i, y_i)$. The parameters α and β are supposed to change into α' and β' , while

$$\begin{aligned}\alpha' &= \xi\alpha + \lambda', \text{ and} \\ \beta' &= \xi\beta + 1\end{aligned}\quad (8)$$

Then, the surprise map with surprise value τ is estimated as the KL-divergence as follows:

$$\tau(x_i, y_i) = KL(p_i(x_i, y_i), p_i(x_i, y_i | D)) \quad (9)$$

The predicted information gain is then quantified as the KL-divergence of the prior and the predicted estimated posterior probability distributions over all the interesting pixels in the surprise map.

P and Q are the normalized prior and the predicted posterior probability mass functions (pmf) with discrete variables: the pixel indexes x_i, y_i .

$$\begin{aligned}
I_{bottom-up} &= KL(P_{bottom-up} \mid \mid Q_{bottom-up}) = \\
&= \sum_{x_i=z_1} \sum_{y_i=z_2} P(x_i, y_i) \cdot \log \frac{P(x_i, y_i)}{Q((x_i, y_i), \hat{\Omega})} \text{ in [bit]}
\end{aligned} \tag{10}$$

where

$$P(x_i, y_i) = \frac{\tau^k(x_i, y_i)}{d^k(x_i, y_i)} \tag{11}$$

$$Q((x_i, y_i), \hat{\Omega}) = \frac{\tau^{k+1k}(x_i, y_i)}{d^{k+1k}((x_i, y_i), \hat{\Omega})} \tag{12}$$

with the surprise value τ of the pixel (x_i, y_i) and the weighting factor d indicating the distance between the pixel (x_i, y_i) and the image centre of the camera lens.

3.5 Context-based combination of top-down and bottom-up attention selections

There are two dominated arts of context recognition approach: the object-based context recognition and the gist-based context recognition. For the object-based context recognition the robot recognizes certain objects as the symbols of certain scenes and adapts its behaviour and tasks to this situation. On the other side, the gist-based context recognition provides the robot a rough idea about what kind of scene the robot is located. In the case that the robot has no previous knowledge about the situation, we consider here only the latter one and try to determine how familiar the current context is and how the robot should adapt its attention selection to this kind of context.

Firstly, we consider the static environment as familiar environment for the robot and the dynamic environment as less familiar environment because of the moving objects causing change and danger. Therefore, we define the context familiarity using motion map histograms computed by three successive input images.

Each normalized histogram of motion map can be regarded as a discrete distribution. Because the perception of human is expectation-based, we calculated the relative entropy s of the histograms of the two consecutive motion maps as the chaos degree of the context. A threshold S should be experimentally specified and applied to determine the weighting factor $\nu(s)$ (see Eq. 2).

3.6 Experiments and results

To evaluate the performance of our view direction planning strategy, the following experiments were conducted. Firstly, four different scenes are investigated to calculate the chaos degree threshold S . Then, experiments in a robot locomotion scenario are conducted, in an environment without surprising event as well as in an environment with a surprising event.

3.6.1 Experiment setup

The experiments are executed in a corridor using ACE (see Fig. 3.4). Four artificial landmarks are installed at the same height as the camera optical axis.



Fig. 3.4. Experiment scenario using ACE and four artificial landmarks

The mobile platform moved straight forward. About every 0.5m a view direction planning is executed and an optimal view direction will be applied for the next 0.5m. We define the view direction Ω as the angle between the locomotion direction and the camera optical axis in the horizontal plane. At the start point (0, 1.25)m the camera has an initial view direction 0° towards the locomotion direction.

3.6.2 Context investigation

Firstly, we specified the chaos degree threshold S . We gathered four different scenes, shown in Fig. 3.5, and calculated their chaos degrees s .

- scene 1: a floor with no moving objects present (Fig. 3.5, column 1)
- scene 2: a square with crowded people (Fig. 3.5, column 2)
- scene 3: a floor with people suddenly appearing (Fig. 3.5, column 3)
- scene 4: a street with a vehicle moving very fast (Fig. 3.5, column 4)

The rows show the consecutive time steps $k-2$, $k-1$ and k at a frame rate of 30pfs. It is obvious that the context almost does not change in the first scene. Therefore, the chaos degree is very small, namely 0.0526. In comparison to the first scene, in the scene 4 the environment changes very much because of the vehicle movement. Hence, the chaos degree in this context is very large, namely 1.1761. For the second scene we have obtained a small chaos degree, namely 0.0473, because the context change is relatively small although there is motion. This context can be regarded as a familiar context, since no surprise exists. In the third scene the chaos degree is large, because a person appeared suddenly in the second image and therefore, the context change is relatively large. For the further experiments we will set S equal 1.0.

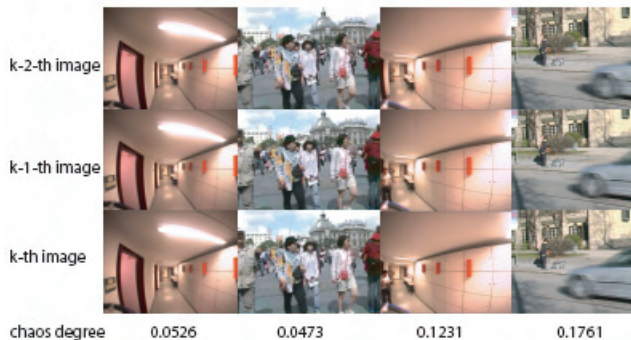


Fig. 3.5. The context chaos degrees in four various scenes

3.6.3 Robot locomotion with and without surprising events

Firstly, the robot moved in a static environment with a constantly low context chaos degree, accomplishing the self-localization task. The image sequence and the respective optimal view directions are shown in Fig. 3.6. If there is no surprising event in the environment, the camera directed its gaze direction to the task-relevant information -- the landmarks (row 1).

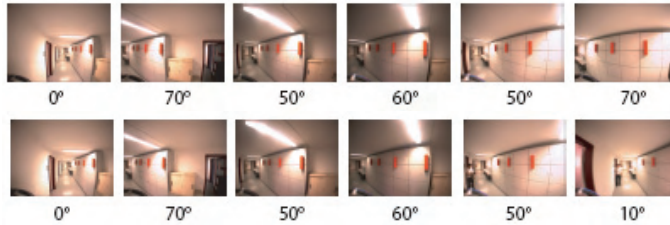


Fig. 3.6. The image sequence and the respective camera view directions in an environment without surprising event (row 1) and with surprising event (row 2)

Fig. 3.6 (row 2) also illustrates an image sequence with the optimal view directions during the locomotion in an environment with surprising event. Most of the time the environment was static with a low context chaos degree (see Fig. 3.5, column 1) and the robot planned its view direction based on the top-down model for the localization task. At the fifth step a person appeared suddenly. Because of the high context chaos degree caused by the surprising event at this moment (see Fig. 3.5, column 3) the camera planned its view direction based on bottom-up attention, tried to locate the surprising event in the environment and changed its view direction from 50° to 10°.

4. GPU aided implementation of bottom-up attention

4.1 Related work

Because of the essential real-time capability of the bottom-up attention, various implementations are proposed. A real-time implementation of the saliency-based model of visual attention on a low power, one board, and highly parallel Single Instruction Multiple Data (SIMD) architecture called Protoeye is proposed in (Ouerhani et al., 2002) in 2002. The implemented attention process runs at a frequency of 14 fps at a resolution of 64×64 pixels. In 2005 another real-time implementation of a selective attention model is proposed (Won et al., 2005). In this model intensity features, edge features, red-green opponent features and blue-yellow opponent features are considered. Their model can perform within 280ms at Pentium-4 2.8GHz with 512MB RAM on an input image of 160×120 pixels.

In the same year a distributed visual attention on a humanoid robot is proposed in (Ude et al., 2005). In this system five different modalities including colour, intensity, edges, stereo and motion are used. The attention processing is distributed on a computer cluster which contains eight PCs. 4 run Windows 2000, 3 Windows XP and 1 Linux. Five of the PCs are equipped with 2x2.2 GHz Intel Xeon processors, two with 2x2.8 GHz Intel Xeon processors, and one with 2 Opteron 250 processors. A frequency of 30 fps with input images with 320×240 pixels is achieved.

In 2006 a GPU based saliency map for high-fidelity selective rendering is proposed (Longhurst et al., 2006). This implementation is also based on the saliency map model proposed in (Itti et al., 1998). In this implementation a motion map and a depth map as well

as habituation are also integrated. However, they use a Sobel filter instead of the complex Gabor filter to produce the orientation maps. No iterative normalization is computed. For an input image at a resolution of 512×512 the saliency map generation takes about 34ms using NVIDIA 6600GT graphics card. No CUDA technology is used.

The most comparable implementation to our implementation is proposed in (Peters & Itti, 2007), because both of them use the same parameter values as those set in (Itti et al., 1998; Walther & Koch, 2006). For a 640×480 colour input image, running in a single-threaded on a GNU/Linux system (Fedora Core 6) with a 2.8GHz Intel Xeon processor, the CPU time required to generate a saliency map is 51.34ms at a precision of floating-point arithmetic and 40.28ms at a precision of integer arithmetic. Computed on a cluster of 48 CPUs a 1.5-2 times better result is achieved. Currently, the fastest computation of saliency map is 37 fps using multi-threaded mode.

4.2 Graphics processing unit (GPU)

In the last few years, the programmable GPUs have become more and more popular. GPU is specialized for compute-intensive, highly parallel computation. Moreover, the CUDA, a new hardware and software architecture issued by NVIDIA in 2007, allows issuing and managing computations on the GPU as a data-parallel computing device without the need of mapping them in a graphics API (CUDA, 2007). It is the only C-language development environment for the GPU.

The saliency map computation consists of compute intensive filtering in different scales, which is nevertheless highly parallelizable. For real-time application we implemented the computation of saliency map on GeForce 8800 (GTX) graphics cards of NVIDIA, which support the CUDA technology. The GeForce 8800 (GTX) consists of 16 multiprocessors which consists of 8 processors each. All the processors in the same multi-processor always execute the same instruction, but with different data. This concept enables a high-gradely parallel computation of a large amount of similar data. The multi-GPU performance is strongly dependent on an efficient usage of the thread-block concept and the different memories.

A. Thread Batching

Programming with CUDA, the GPU is called *compute device*. It contains a large amount of threads which can execute an instruction set on the device with different data in parallel. A function which is compiled to those instruction set is called *kernel*. In comparison with GPU, the main CPU is called *host*. The goal is to execute the data-parallel and compute-intensive portions of applications on the GPU instead of on the CPU.

Fig. 4.1 shows the thread batching model of the GPU. For each *kernel* function the GPU is configured with a number of threads and blocks. The respective grid of a *kernel* consists of two dimensional blocks. Each block contains up to 512 threads. The input data are divided into the threads. All the threads in a grid execute the same kernel functions. With the thread index *threadIdx* and the block index *blockIdx* we know which data will be processed in which thread. With this structure an easy programming and a good scalability are realized.

B. Memory

The memory access is also a focus for an efficient programming on GPU. There are six different memories in GPU:

- read-write per-thread registers

- read-write per-thread local memory
- read-write per-block shared memory
- read-write per-grid global memory
- read-only per-grid constant memory
- read-only per-grid texture memory

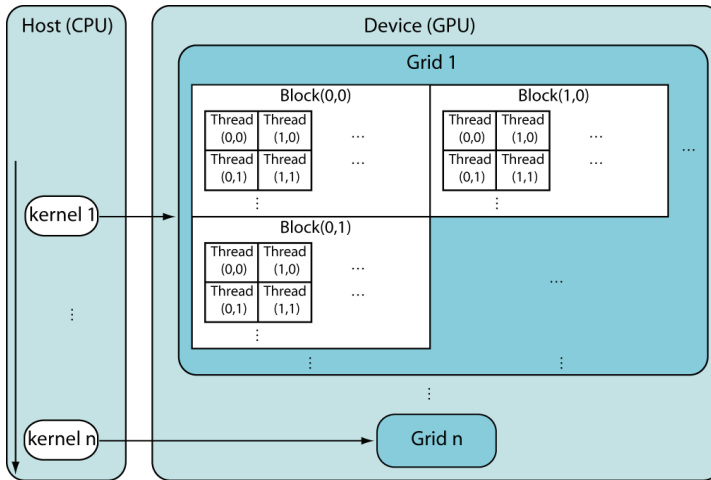


Fig. 4.1. The thread batching model of GPU

Above all, the shared memory and the texture memory are cached, while the read or write access in the not cached global memory always takes 400-600 clock cycles. Only the texture memory and the global memory can be used for a large amount of data. Moreover, the texture memory is optimized for 2D spatial locality and supports many operations such as interpolation, clamping, data type conversion etc.. However, the texture is read-only. The results must be saved in the global memory, which requires data copy between memories.

4.3 Multi-GPU implementation details

In Fig. 4.2 a data flow diagram of our GPU-implementation is illustrated. After an initialization, an input image is firstly converted into 32-Bit floating point such that a high accuracy and a high efficiency will be achieved in the following computation phases. The Gaussian dyadic pyramid is created in the shared memory together with the generation of intensity maps (I-maps), opponent red-green (RG-maps) and blue-yellow maps (BY-maps). We use the Gabor filter to calculate the Orientation-maps (O-maps). The Gabor filter kernel is firstly calculated in the CPU. To spare computational cost, the convolution of the subsampled images with Gabor filter in the space domain is displaced by the multiplication in the frequency domain using Fast Fourier Transform (FFT). Here we conducted a Cuda-image which contains all the images to be filtered by the in the initialization transformed Gabor filter such that only one FFT and eight IFFT are needed for the convolution. The images should be assembled before the transformation and disassembled after the transformation in the texture memory. After that, 9 I-maps, 18 C-maps and 36 O-maps are generated.

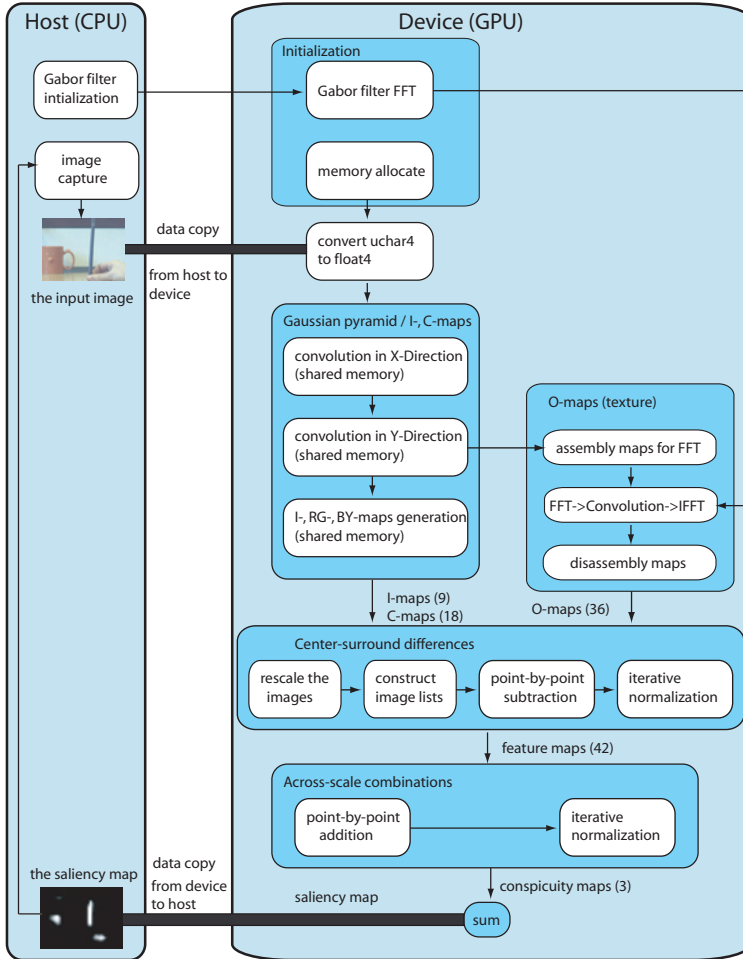


Fig. 4.2. Data flow diagram for GPU-implementation of the saliency map computation

Furthermore, to ease the center-surround differences and the cross-scale combinations, the available maps at different scales are rescaled into the same size. A point-to-point subtraction followed by an iterative normalization is calculated. On the resulting 42 feature maps a point-to-point addition and its following normalization are executed. One conspicuity map in each channel is obtained. At the end, a summation of the conspicuity maps into the saliency map is completed. The detailed description is as follows:

A. Initialization

Firstly, the GPU should be initialized. For the reason that the memory allocation in GPU takes very long, the memory is firstly allocated for different images such as the input images, the images in the Gaussian dyadic pyramids, the feature maps, the conspicuity maps, the rescaled feature and conspicuity maps at the same size as well as the final saliency map.

Since the filter kernel will not be changed during the saliency map computation, we also calculate the Gabor filter in the initialization phase in the CPU and then transform it into the frequency domain. The implementation of the Gabor filter and the FFT-transformation of the Gabor filter will be described in Section 4.3-E in detail.

B. Data type conversion

The input image has the resolution of 640×480 and three 8-bit channels, namely red, green and blue. The image data are copied from the CPU into the global memory of the GPU. Since the global memory is not cached, it is essential to follow the right access pattern to get maximum memory bandwidth. The data type must be such that $sizeof(type)$ is equal to 4, 8, or 16 and the variables of type *type* must be aligned to $sizeof(type)$ bytes (CUDA, 2007). If the alignment requirement is not fulfilled, the accesses to device memory are very costly. The image width fulfills the alignment requirement, while the data amount of each pixel is $3 \times 8 = 24$ Bytes which does not fulfill the alignment requirement. Therefore, we must extend the pixel width with *padding* and insert an extra 8-bit channel (see Fig. 4.3).

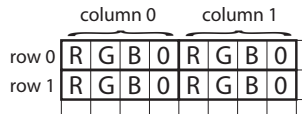


Fig. 4.3. Image data padding

After the *padding* we convert the image data type with *uchar4* into *float4* to achieve the high precision for the following computation using the implicit type conversion of the texture.

C. Gaussian dyadic pyramid computation

In (Walther & Koch, 2006) a 6×6 separable Gaussian kernel $[1\ 5\ 10\ 10\ 5\ 1]/32$ is used for the image size reduction. A two-dimensional convolution contains $6 \times 6 = 36$ multiplications for each output pixel, while a convolution with separable filters only requires $6 + 6 = 12$ multiplications for each output pixel. Therefore, we separate the Gaussian dyadic pyramid computation into two convolutions: one convolution in the horizontal direction to reduce the horizontal dimension, and one convolution in the vertical direction, respectively.

Since each access in the uncached global memory takes 400-600 clock cycles, it is necessary to compute the convolutions in the faster texture memory or shared memory. Bounding the images to a texture requires the data copy between the global memory and the texture memory. Moreover, the data are only readable by kernels through texture fetching. It is more costly than loading the data into the shared memory and compute the convolution there. Therefore, the convolution is computed in the shared memory.

For the convolution in the horizontal direction, the thread and block number are so specified that a block consists of as many threads as the number of the output image columns and a grid has as many blocks as the number of the output image rows. For example, for the subsampling from an input image at 640×480 into an output image at 320×480 , each block has 320 threads, while each grid has 480 blocks. Each thread computes only one pixel in the output image.

Attention must be paid to the threads synchronization, because the convolution in the thread *n* is dependent on the pixels loaded by thread *n-1* and *n+1*.

To deal with the convolution on the image border, we use $[10\ 10\ 5\ 1]/26$ on the left border and $[1\ 5\ 10\ 10]/26$ on the right border.

After that, a following subsampling in the vertical direction can be similarly solved. The input image at 640×480 is subsampled into 8 other scales: 320×240 (scale $\sigma = 1$), 160×120 ($\sigma = 2$), ..., 2×1 ($\sigma = 8$).

D. C-maps and I-maps computation

In the saliency map computation the I-, RG- and BY-maps are required (Walther & Koch, 2006). To make the computation more efficient, we integrate the computation of the I-maps and the C-maps into the Gaussian filter convolution in the vertical direction, because the image data are already in the shared memory after the convolution. Thus, we can spare the time for loading the data from the global memory.

E. O-maps computation

1) *Gabor filter*: To compute the O-maps in different scales, the Gabor filter truncated to 19×19 pixels is used (Walther & Koch, 2006). The Gabor filter is formulated as follows:

$$G_{\psi}(x, y, \theta) = \exp\left(\frac{x'^2 + \gamma^2 y'^2}{2\delta^2}\right) \cdot \cos\left(2\pi \frac{x'}{\lambda} + \psi\right) \quad (13)$$

With

$$x' = x \cos(\theta) + y \sin(\theta), \quad y' = -x \sin(\theta) + y \cos(\theta) \quad (14)$$

(x, y) is the pixel coordinate in the Gabor filter. The parameter values of our implementation are according to (Walther & Koch, 2006). γ stands for the aspect ratio with the value 1, while λ is the wavelength and has the value of 7 pixels. The standard deviation δ is equal $7/3$ pixels, and $\psi \in \{0, \frac{\pi}{2}\}$. θ stands for the orientation angles with $\theta \in \{0^\circ, 45^\circ, 90^\circ, 135^\circ\}$.

As defined in Eq. 14, a Gabor filter consists of a combination of a 2D Gaussian bell-shaped curve and a sine ($\psi = \pi/2$) and cosine function ($\psi = 0$). In each direction, the image should be filtered twice and summed as follows:

$$M_{\theta}(\sigma) = \|M_I(\sigma) * G_0(\theta)\| + \|M_I(\sigma) * G_{\pi/2}(\theta)\| \quad (15)$$

with $M_I(\sigma)$ the I-Maps at scale σ .

2) *FFT and IFFT*: Since a convolution with the 19×19 Gabor filter is too costly, we use FFT and IFFT to accelerate this process significantly. The Gabor filter and the to be convoluted images should be converted into the frequency domain using FFT at first, and multiplied with each other. Then, the result is converted from the frequency domain into the space domain using IFFT. In doing this, the complexity sinks from $O(n^4)$ (2D convolution) to $O(n^2 \log n)$ (2D FFT).

As mentioned in 4.3-A, the FFT of the Gabor filter should be computed in the initialization, because it will never be modified in the saliency map generation. Using CUFFT library (CUDA CUFFT, 2007) we compute from the original Gabor filter eight FFTs with four different orientations and two different forms (sine and cosine).

Due to the fact that the input image (640×480) and the subsampled image at scale 1 (320×240) are not used for the following saliency map computation, $7 \times 4 \times 2 = 56$ convolutions for

the O-maps are needed (7 scales, 4 orientations and 2 forms). We assembly the images in 7 scales together into an Cuda-image (see Fig. 4.5, left) such that just 1 FFT and 8 IFFTs instead of 7 FFT and 56 IFFTs are computed. For an input image with 640 x 480 pixels, an image with 256 x 256 is big enough to have all the images into itself.

Using the texture a modus named "clamp-to-border" is supported, which makes the image copy very simple. If a pixel outside the texture border is accessed, this pixel has the same color as the border. Therefore, instead of copying the pixel from (0, 0) to (n-1, n-1), we copy the pixel from (-9, -9) to (n+8, n+8) of an image with n x n pixels. In doing this we get the border extension for the convolutions.

Before we compute the FFT of the Gabor filter, we should resize the Gabor filter kernel (19 x 19) into the same size as the to be convoluted image (256 x 256), because the convolution using FFT only can be applied on the input data of the same size (Podlozhnyuk, 2007). The expansion of the Gabor filter kernel to the image size should be executed as shown in Fig. 4.5 right: cyclically shift the original filter kernel such that the kernel center is at (0, 0).

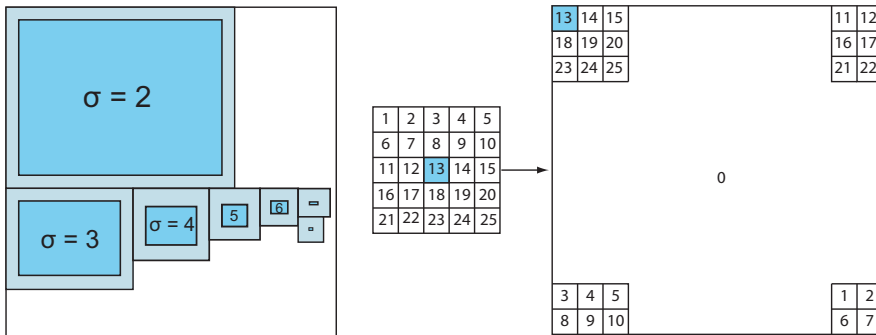


Fig. 4.5. The image (left) and the filter kernel (right) prepared for FFT

In the center-surround differences, 6 feature maps in the intensity channel, 12 feature maps in the color channel and 24 feature maps in the orientation channel are computed between the selected fine scale maps and the coarse maps. To execute this subtraction, the images should be enlarged or reduced into the same size and then a point-by-point subtraction is accomplished. We reduce the images at scale 2 and 3 into scale 4 and enlarge the images at scale 5, 6, 7, 8 also into scale 4. At the end all the images are at scale 4 and have 40 x 30 pixels. For those enlargements and reductions we use the texture concept again by bounding them to the textures.

	I-maps						O-maps						
list center	2	2	3	3	4	4	...	2	2	3	3	4	4
list surround	5	6	6	7	7	8	...	5	6	6	7	7	8
list difference	2-5	2-6	3-6	3-7	4-7	4-8	...	2-5	2-6	3-6	3-7	4-7	4-8

Fig. 4.6. The image lists configuration

Since the images are rescaled into 40 x 30 pixel at this step, we construct three lists to make the computation as parallely as possible. Fig. 4.6 shows the configuration of the lists. Each list contains 6 x 7 = 42 images with different scale number (but in the same size 40 x 30) and

channels. The threads and blocks are so parametrized that 42 blocks are configured. Each block is responsible for one image in the list. 42 images are processed in only one kernel function parallelly. This list-concept is also used for the iterative normalization and the cross-scale combinations.

G. Iterative normalization

The iterative normalization $N(\cdot)$ is an important component in the whole computation. It simulates local competition between neighboring salient locations (Itti et al., 1998). Each iteration contains self-excitation and neighbor-induced inhibition, which can be implemented using a difference of Gaussian filter (DoG) (Itti & Koch, 1999):

$$DoG(x, y) = \frac{c_{ex}^2}{2\pi\sigma_{ex}^2} e^{-\frac{x^2+y^2}{2\pi\sigma_{ex}^2}} - \frac{c_{inh}^2}{2\pi\sigma_{inh}^2} e^{-\frac{x^2+y^2}{2\pi\sigma_{inh}^2}} \quad (16)$$

with $\sigma_{ex} = 2\%$ and $\sigma_{inh} = 25\%$ of the input image width, $c_{ex} = 0.5$, $c_{inh} = 1.5$ and the constant inhibitory term $C_{inh} = 0.02$. At each iteration the given image M is computed as follows (Itti & Koch, 1999):

$$M \leftarrow |M + M * DoG - C_{inh}|_{\geq 0} \quad (17)$$

The inseparable DoG filter is divided into two separable convolution filters, one Gaussian filter for excitation with 5×5 pixels and one Gaussian filter for inhibition with 29×29 pixels for an input image at 40×30 . The larger the input image is, the bigger are the filter kernels. The kernel size can be computed as follows:

$$size_{(ex|inh)} = 2 \cdot floor(\sigma_{(ex|inh)} \cdot \sqrt{-2 \cdot \ln(1/100)}) + 1 \quad (18)$$

The 153 iterations on 51 images are very costly. Although the shared memory size is limited, the images at 40×30 and the respective filter kernels (4916 Byte) can fit into it. In doing this, a 10 times acceleration is obtained, whereas the lists mentioned in 4.3-F are also used.

H. Combination into the saliency map

In the following cross-scale combinations no image rescaling is needed. It is only a question of point-by-point integration of the feature maps into conspicuity maps \bar{I} , \bar{C} and \bar{O} . The saliency map is a linear combination of the normalized conspicuity maps.

I. Multi-GPU utilization

A parallel utilization of multi-GPU enables a significant acceleration of the saliency map computation. To avoid the intricateness of a multi-process mode, a multi-threaded mode is used to manage the multi-GPU utilization. In a multi-threaded mode, in addition to a main thread several threads are utilized. Each thread is responsible for one GPU. Fig. 4.7 illustrates the multi-threaded mode in a petri-net. Two semaphores are used to ensure the synchronization of the threads. The semaphore 1 sends a signal to the main thread if one or more GPUs are idle and is initialized with the number of the applied GPUs. The semaphore 2 starts one of the GPU-threads. Interestingly, in the main thread, at $t_{0,4}$ a thread is started, while at $t_{0,5}$ a saliency map is ready to be taken. Using this multi-threaded mode the frame rate can be significantly increased.

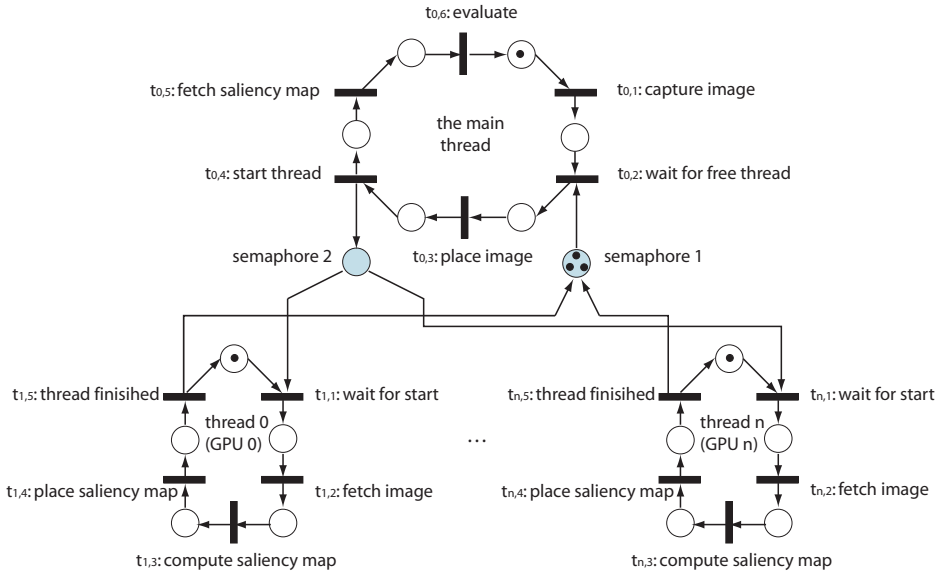


Fig. 4.7. The petri-net structure for multi-threaded mode

4.4 Results and discussion

We tested our multi-GPU implementation using 1 to 4 NVIDIA GeForce 8800 (GTX) graphics cards. The computers are equipped with different CPUs and 64-bit linux systems. The computational time is the average processing time of 1000 input images at a resolution of 640×480 pixels.

Tab. 4.1 shows the detailed processing time protocol. The most costly step is the initialization which has a computational time of 328ms. The memory allocation happens only once and needs almost 50MB RAM. The saliency map computation takes only about 10.6ms with a frame rate of 94.3 fps, respectively. In the GFLOPS performance estimation, only the floating-point operations are considered. The address-pointer-arithmetic, the starting of the CUDA functions and the memory copy accesses, which are very time-consuming and have, therefore, a strong influence on the computational time, are not considered.

Saliency map computation	Time	FLOP	GFLOPS
initialization	328ms		
Gaussian pyramid I-, C-maps	2,10ms	6.482.049	3,09
FFT, convolution, IFFT	2,39ms	27.867.923	11,66
image rescaling	0,89ms	294.000	0,33
center-surround differences	0,16ms	151.200	0,95
iterative normalization	4,74ms	34.876.690	7,36
Integration into saliency maps	0,33ms	62.390	0,19
total	10,61ms	69.734.252	6,57

Table 4.1. Computational time registration using 1 GPU

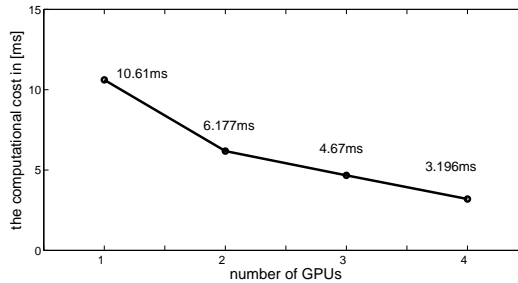


Fig. 4.8. Comparison of computational time using 1 to 4 GPUs

Fig. 4.8 illustrates the computational time using 1 to 4 GPUs, which shows a very good scalability of the multi-GPU implementation.

In Tab. 4.2 the performance of the iLab's implementation (Peters & Itti, 2007) and our implementation is compared. Working on the images with the same resolution and the same precision, iLab uses the 2.8GHz Intel Xeon processor and achieves a frequency of 19.48 Hz, while using our implementation a frequency of 313 Hz is obtained. Using multi-threaded mode, the maximum speed of iLab is 37 fps which is still about 8.5 times slower than our implementation.

	iLab's implementation	our implementation
resolution	640 x 480	640 x 480
hardware	2.8GHz Intel Xeon processor	4 NVIDIA GeForce 8800 (GTX)
precision	floating-point	floating-point
computational time	51.34ms	3.196ms
frequency	19.48 Hz	313 Hz

Table 4.2. Comparison between iLab's implementation and our implementation

5. Visual odometry for ACE

The goal of ACE is to navigate in an unpredictable and unstructured urban environment. For achieving the aim, accurate pose estimation is one of the preconditions. As humans, we use visual information to estimate the relative motion between ourselves and a reference object. If we close our eyes, we can still estimate the motion by feeling the foot step. Even if we move in a car and close our eyes, we can use inertial sensor in the body, such as inner ear, to tell how our motion is. By now, ACE only has the information from the angle-encoders on the wheels. If there are sands, cobblestone on the ground, the wheels will slip, which causes an inaccurate localization. Therefore, we want to use the visual information to support the localization. We mount a high-speed camera in the front of ACE. The camera looks straight towards the ground.

In this section a visual odometry system is presented to estimate the current position and orientation of ACE platform. The existing algorithms of optical flow computation are analyzed, compared and an improved sum-of-absolute difference (SAD) algorithm with high-speed performance is selected to estimate the camera ego-motion. The kinematics model describing the motion of ACE robot is set up and implemented. Finally the whole odometry system was evaluated within appropriate scenarios.

5.1 Background

How to locate the position and orientation of a moving object has long been a research focus of the computer vision community. The probably existing problems could be being robust against complicated environment, different ground situations and changing brightness. Most visual odometry methods include three phases: firstly, a suitable optical flow computation algorithm should be selected to determine the optical flows in a series of successive images. Then, the translation and rotation of the camera should be estimated according to these optical flows acquired in the first step. At last, a geometry model denoting the relation between camera and robot should be established so that the localization of the robot can be deduced from the position of the camera.

5.1.1 Optical flow techniques

The computation of optical flows has been a key problem discussed in the processing of image sequences for many years (Barron et al., 1994). Nowadays there are two most popular techniques: Matching-based method and differential method. Block-based matching is applied in many aspects of computer vision area. It's also one of the most important techniques for optical flow computation. Two simple algorithms, sum-of-absolute difference (SAD) and sum-of-squared difference (SSD), are usually used to find the best match. They are more efficient than the other techniques. Lucas & Kanade is a typical and classical differential technique, which is based on the gradient constraint. It has a comparative robustness and accuracy in the presence of noise and is feasible in reality.

5.1.2 Pose estimation using optical flow information

Pose estimation is the procedure to compute the position and orientation of a camera relative to the world coordinate. Using image Jacobian matrix, the relationship between object velocity in 3-D world and its image-plane velocity on the image sensor is described as follows:

$$\begin{bmatrix} \dot{u} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} \frac{\lambda}{z} & 0 & -\frac{u}{z} & -\frac{uv}{\lambda} & \frac{\lambda^2 + u^2}{\lambda} & -v \\ 0 & \frac{\lambda}{z} & -\frac{v}{z} & \frac{-\lambda^2 - u^2}{\lambda} & \frac{uv}{\lambda} & u \end{bmatrix} \begin{bmatrix} T_x \\ T_y \\ T_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \quad (19)$$

where T_x, T_y, T_z are translation velocities of the camera in world coordinate in three directions, $\omega_x, \omega_y, \omega_z$ angular velocities in three directions. \dot{u} and \dot{v} are the pixel velocity along x and y directions in image plane, while u and v are the corresponding pixel coordinates in image plane. Normally we have more than 3 feature points on the image plane, so the equation system is redundant.

5.1.3 Related work

Jason Campbell et al. (Campbell et al. 2005) designed a model using monocular camera mounted at the robot's front and viewing front and underside of the ground. The flow

vectors are divided into three parts: a dead zone near the horizon is defined and discarded in the computational processing; the vectors above the horizon are used to calculate the robot rotation while the vectors below the horizon are used to estimate the robot translation. Similar to the model established by Campbell, the monocular camera in Wang's model (Wang et al. 2005) focuses only on the locally planar ground, and calculates the translation and rotation together. Both of the models use Lucas & Kanade method to obtain the optical flow. Utilizing the Harris corners detection and normalized correlation, Nister presented a system (Nister et al. 2004) that provides an accurate estimation but works relative slowly. In Fernandez's work (Fernandez & Price 2004), utilizing the task-sharing abilities of the operating system, the problem of synchronization across multiple frame-grabbers can be solved. In order to have a better efficiency, the SAD algorithm is used here. In Dornhege's work (Dornhege & Kleiner, 2006) the salient features are tracked continuously over multiple images and then the differences between features that denotes the robot's motion are computed. An inertial measurement unit (IMU) is employed here to estimate the orientation.

5.2 Hardware and modeling

5.2.1 Hardware description

Fig. 5.1 illustrates the hardware configuration. A high-speed camera and a 1394b PCI-express adapter are used in our system to capture and transfer the images to the computational units. The dragonfly® express camera (Point Grey Research Inc.), fitted with a normal wide-angle lens, can work at 200 fps with the resolution of 640x480 pixels. Utilizing the IEEE-1394B (Firewire 800) interface, the camera is connected to our vision processing computer with an AMD Phenom 9500 @2.2GHz Quad-Core processor and 4 GB memory.



Fig. 5.1. Hardware configuration

5.2.2 Kinematics modeling

Because ACE will explore the outdoor urban environments, e.g. the city centre of Munich, and communicate frequently with the humans, so the camera for visual odometry may not gaze directly forward. For avoiding the disturbance of moving crowd, the camera is mounted in the front of ACE and the optical axis is perpendicular to the ground.

The camera is fixed on ACE such as represented in Fig. 5.2. The relative position between the camera and the robot does not change in the whole process. Any actuated motion of the robot will result in a movement of the camera relative to its original position. Because the displacement between camera and ground in z-direction is much smaller than the distance between camera and ground z , we can approximately assume that the ground is a flat plane

and the ACE-platform displaces without any roll and pitch angle. Based on this assumption only 3 variables must be considered: the movements in x and y directions and the orientation around the z axis. We divide the movement of robot in two parts (see Fig. 5.3).

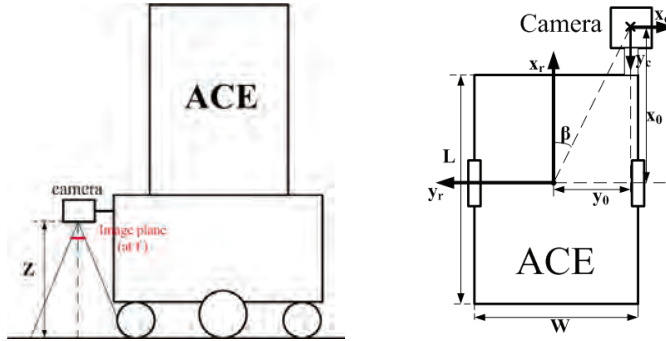


Fig. 5.2. Cutaway and planform of the visual odometry configuration

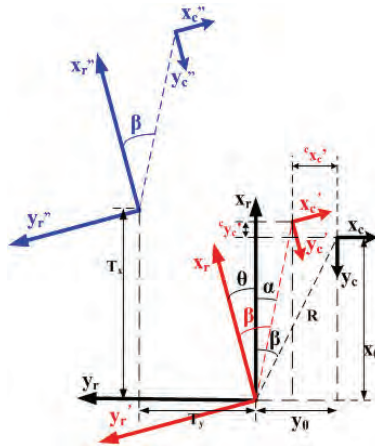


Fig. 5.3. Geometry relationship between robot and camera in motion with frames and variables definition

Firstly, it rotates with an angle of θ without any translation. Then, the robot has movement of (T_x, T_y) . After that, the three variables of camera relative to its original position can be denoted as:

$$\begin{aligned}
 \theta_c &= \theta \\
 X_c &= y_0 - R \cdot \sin(\beta + \theta_c) - T_y \\
 Y_c &= x_0 - R \cdot \cos(\beta + \theta_c) - T_x
 \end{aligned}
 \tag{20}$$

where $R = \sqrt{x_0^2 + y_0^2}$ and $\beta = \arctan(\frac{y_0}{x_0})$.

5.3 Motion estimation

Our long-term objective is to fuse the visual information at 200 Hz and the information provided by the angle-encoders at 30 Hz to achieve a high accuracy visual odometry. Currently, we focus on the visual information. The vision processing is as follows: the input images will be undistorted at first. Then, using SAD the optical flow is computed. The relationship between optical flow and the camera ego-motion is indicated by image Jacobian. To reduce the noise and optimize the results of the redundant equations, a Kalman filter is applied.

5.3.1 Optical flow algorithm – elaborated SAD

Compared with other optical flow computation algorithms, SAD performs more efficiently and less system resources are required. The size of our images is 640x480 pixels and the central 400x400 pixels are chosen as interest area. A searching window of 20x20 is defined so there are totally 400 windows in this interest area. SAD algorithm is used in every window with a block size of 8x8 pixels. This block is regarded as original block in frame n-1 and compared with the corresponding neighbour blocks within the searching window in frame n. The block with the least SAD values in frame n will be taken as the matching block. The distance between the original block and the matching block is defined as optical flow value of this searching window. After SAD matching 400 sets of optical flow values have been acquired and a further elaboration is fulfilled as follows: The searching windows on the boundary of the interest area are abandoned and the remaining 18x18 windows can be separated into 36 groups. Each group consists of 3x3 windows as show in Fig. 5.4. In every group we set a limit to eliminate some windows whose optical flow values seem not to be ideal enough. The average optical flow values of remaining windows in every group should be determined and could be seen as a valid optical flow value of this group. Every group can be considered as a single point and we just calculate the optical flow values of 36 feature points with a better accuracy.

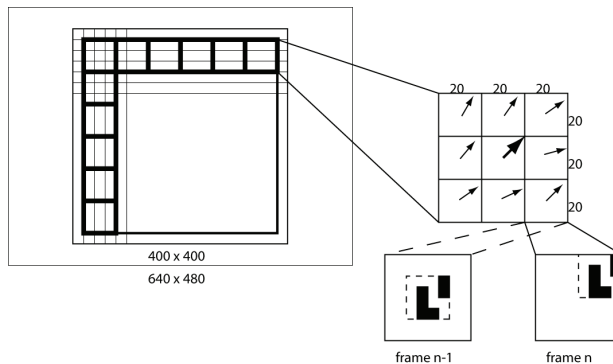


Fig. 5.4. Elaborated SAD algorithm

5.3.2 State estimation – Kalman filter

After calculating optical flow values with an elaborated SAD algorithm, we apply Kalman filter to determine the redundant equations based on image Jacobian matrix. The basic thought of Kalman filter is to predict the state vector x_k according to the measurement

vector z_k . Based on the assumption we have made in kinematics model, only T_x , T_y and ω_z are required and therefore the state vector is composed of only three elements. The measurement vector z_k comprises the 36 sets of points velocities acquired from optical flow values of 36 feature points. The measurement matrix is a simplified image Jacobian matrix J , the redundant equations can be described as follows:

$$\begin{bmatrix} \dot{u}_1 \\ \dot{v}_1 \\ \vdots \\ \dot{u}_{36} \\ \dot{v}_{36} \end{bmatrix} = \begin{bmatrix} \frac{\lambda}{z} & 0 & -v_1 \\ 0 & \frac{\lambda}{z} & u_1 \\ \vdots & \vdots & \vdots \\ \frac{\lambda}{z} & 0 & -v_{36} \\ 0 & \frac{\lambda}{z} & u_{36} \end{bmatrix} \cdot \begin{bmatrix} T_x \\ T_y \\ \omega_z \end{bmatrix} \tag{21}$$

The basic process of Kalman filter in our experiment is as follows:

$$\begin{aligned} x_k &= x_{k-1} + w_k \\ z_k &= J \cdot x_{k-1} + v_k \end{aligned} \tag{22}$$

Random variables w_{k-1} and v_k represent the process noise and measurement noise respectively. The estimation process can be divided into two parts: predict part and correct part. At the beginning, the camera velocity vector, which is also the state vector in Kalman filter, is initialized with null vector, after the predict part, prior camera velocity estimation and prior error covariance estimation are transferred to the correct part. In correct part the posterior camera velocity estimation are computed by incorporating current point velocity vector, which is also the measurement vector. A posterior error covariance is also calculated in correct part and together with posterior camera velocity estimate transferred as initialization of the next step. In every step the posterior camera velocity estimation is the result of the redundant equations.

5.4 Experiments results

In the ACE platform there is an encoder which can estimate the current position of ACE. We read the data from the encoder at a frequency of 4-5Hz and consider them as ground truth. The camera mounted on ACE works at a frequency of 200Hz. Our experiment data is obtained when ACE is moving in the environment of stone sidewalk. The experiment is divided into two parts. In the first part, ACE ran about 6,7m in a straight line, which is taken as pure translation. The second part is pure rotation test. ACE only rotated at the starting point and passes about 460 grads. Two series of images are captured and saved, and then the experiment is carried out offline. The motion estimation computation works also at 200Hz. Fig. 5.5 left shows the results of estimating the robot displacements in pure translation. The red curve indicates the displacement in x-direction measured by encoder, and the blue curve indicates the displacement in x-direction estimated by visual odometry. The right part of Fig. 5.5 shows the angular result in pure rotation. The red curve indicates the ground truth from encoder, and the black curve indicates the estimation result from visual odometry.

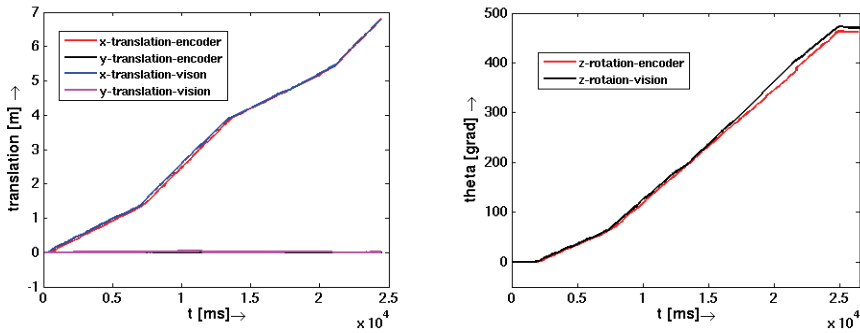


Fig. 5.5. Position estimation in pure translation (left) and in pure rotation (right)

The right part of Fig. 5.5 shows the angular result in pure rotation. The red curve indicates the ground truth from encoder, and the black curve indicates the estimation result from visual odometry.

6. Conclusions and future work

In this chapter, two high-speed vision systems are introduced, which can acquire and process visual information in real time and are used for the visual attention and navigation of the Autonomous City Explorer (ACE).

An information-based view direction planning is proposed to rapidly detect the surprising event in the environment during accomplishing predefined tasks. This high performance is facilitated and ensured by high-speed cameras and high-speed processors such as Graphics Processing Units (GPUs). A frequency of 313 fps on input images at 640×480 pixels is achieved for the bottom-up attention computation, which is about 8.5 times faster than the standard implementation on CPUs. For the high speed visual odometry, our algorithm performs well according to the experiments results. The time delay of close-loop control can be decreased and the system stability can be improved.

Further development based on these two high-performance vision systems is planned to improve the self-localization and navigation accuracy. Besides, a suitable data fusion algorithm should be selected to combine data from encoder and visual. The visual attention system should also be extended for the application of human-robot interaction.

7. Acknowledgment

This work is supported in part within the DFG excellence initiative research cluster *Cognition for Technical Systems – CoTeSys*, see also www.cotesys.org.

8. References

- Barron, J.; Fleet, D. & Beauchemin, S. (1994). Performance of optical flow techniques, *International journal of computer vision*, Vol.12, No. 1, February 1994, pp. 43-77, ISSN: 0920-5691.
- Campbell, J.; Sukthankar, R.; Nourbakhsh, I. & Pahwa, A. (2005). A robust visual odometry and precipice detection system using consumer-grade monocular version,

- Proceedings of the 2005 IEEE International Conference on robotics and automation*, pp.3421 – 3427, ISBN: 0-7803-8915-8 , Barcelona, April 2005
- CUDA (2007). *Programming Guide Version 1.1*. NVIDIA
- CUDA CUFFT Library (2007). NVIDIA
- Davison, A.J. (1998). *Mobile Robot Navigation Using Active Vision*. PhD thesis. Robotics Research Group, Department of Engineering Science, University of Oxford
- Dornhege, C. & Kleiner, A. (2006). Visual odometry for tracked vehicles, *Proceeding of the IEEE international workshop on safety, security and rescue robotics*, Gaithersburg, USA, 2006
- Fernandez, D. & Price, A. (2004). Visual odometry for an outdoor mobile robot, *Proceeding of the 2004 IEEE conference on robotics and mechatronics*, Singapore, December 2006
- Frintrop, S. (2006). *VOCUS: A Visual Attention System for Object Detection and Goal-directed search*. PhD thesis
- Im, S. & Cho, S. (2006). Context-Based Scene Recognition Using Bayesian Networks with Scale-Invariant Feature Transform, *ACIVS 2006*, LNCS 4179, pp. 1080-1087
- Itti, L.; Koch, C & Niebur, E. (1998). A model of saliency-based visual attention for rapid scene analysis, *Pattern Analysis and Machine Intelligence*, vol. 20, pp. 1254-1259
- Itti, L. & Koch, C. (1999). A comparison of feature combination strategies for saliency-based visual attention systems, *SPIE human vision and electronic imaging IV (HVEI'99)*, San Jose, CA pp 473-482
- Itti, L. & Baldi, P. (2005). A Principled Approach to Detecting Surprising Events in Video, *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, June, 2005
- Kühnlenz, K. (2006a). *Aspects of Multi-Focal Vision*, PhD thesis, Technische Universität München.
- Kühnlenz, K., Bachmayer, M. and Buss, M. (2006b). A Multi-Focal High-Performance Vision System, *Proceedings of the International Conference of Robotics and Automation (ICRA)*, pp. 150-155, Orlando, USA, May 2006.
- Lidoris, G., Klasing, K., Bauer, A., Xu, T., Kühnlenz, K., Wollherr, D. & Buss, M. (2007). The Autonomous City Explorer Project: Aims and System Overview, *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2007.
- Longhurst, P.; Debattista, K. & Chalmers, A. (2006). A GPU based Saliency Map for High-Fidelity Selective Rendering, *AFRIGRAPH 2006*, Cape Town, South Africa
- Nister, D., Naroditsky, O., Bergen, J. (2004). Visual odometry, *Proceedings of the 2004 IEEE computer society conference on computer Vision and Pattern Recognition*, Vol.1, pp. 652-659, ISBN: 0-7695-2158-4, Washington DC, July 2004
- Ouerhani, N.; Hügli, H.; Burgi, P.Y. & Ruedi, P.F. (2002). A Real Time Implementation of the Saliency-Based Model of Visual Attention on a SIMD Architecture, *DAGM 2003*, LNCS 2449, pp. 282-289
- Ouerhani, N.; Hügli, H.; Gruener, G. & Codourey, A. (2005). A Visual Attention-Based Approach for Automatic Landmark Selection and Recognition, *WAPCV 2004*, LNCS 3368, pp. 183-195
- Pellkofer, M. & Dickmanns, E.D. (2000). EMS-Vision: Gaze Control in Autonomous Vehicles, *Proceedings of the IEEE Intelligent Vehicles Symposium 2000*, Dearborn, USA

- Peters, R.J. & Itti, L. (2007). Applying computational tools to predict gaze direction in interactive visual environments, *ACM Transactions on Applied Perception*, May 2007, Pages 1-21
- Podlozhnyuk, V. (2007). *FFT-based 2D convolution*. NVIDIA
- Remazeilles, A. & Chaumette, F. (2006). Image-based robot navigation from an image memory, *Robotics and Autonomous Systems*
- Seara, J.F. & Schmidt, G. (2005). Gaze Control Strategy for Vision-Guided Humanoid Walking, *at-Automatisierungstechnik*, vol. 2, pp. 49-58
- Torralba, A. & Sinha, P. (2001). Statistical Context Priming for Object Detection. Proceedings of the International Conference on Computer Vision (ICCV), pp. 763-770, Vancouver, Canada
- Ude, A.; Wyart, V.; Lin, L.H. & Cheng, G. (2005). Distributed Visual Attention on a Humanoid Robot, *Proceedings of 2005 5-th IEEE-RAS International Conference on Humanoid Robots*
- Walther, D. & Koch, C. (2006). Modeling attention to salient proto-objects, *Science Direct, Neural Networks*, vol. 19, pp. 1395-1407
- Wang, H.; Yuan, K.; Zou, W. & Zhou, Q. (2005). Visual odometry based on locally planar ground assumption, *Proceeding of the 2005 IEEE international conference on information acquisition*, Hong Kong and Macau, China, June 2005
- Won, W. J.; Ban, S. W. & Lee, M. (2005). Real Time Implementation of a Selective Attention Model for the Intelligent Robot with Autonomous Mental Development, *IEEE ISIE 2005*, June 20-23, Dubrovnik, Croatia