J Real-Time Image Proc (2011) 6:171–186
DOI 10.1007/s11554-010-0156-7

SPECIAL ISSUE

Bayesian real-time perception algorithms on GPU

Real-time implementation of Bayesian models for multimodal perception using CUDA

João Filipe Ferreira · Jorge Lobo · Jorge Dias

Received: 24 September 2009 / Accepted: 2 February 2010 / Published online: 26 February 2010

Abstract In this text we present the real-time implementation of a Bayesian framework for robotic multisensory perception on a graphics processing unit (GPU) using the Compute Unified Device Architecture (CUDA). As an additional objective, we intend to show the benefits of parallel computing for similar problems (i.e. probabilistic grid-based frameworks), and the user-friendly nature of CUDA as a programming tool. Inspired by the study of biological systems, several Bayesian inference algorithms for artificial perception have been proposed. Their high computational cost has been a prohibitory factor for real-time implementations. However in some cases the bottleneck is in the large data structures involved, rather than the Bayesian inference per se. We will demonstrate that the SIMD (single-instruction, multiple-data) features of GPUs provide a means for taking a complicated framework of relatively simple and highly parallelisable algorithms operating on large data structures, which might take up to several minutes of execution with a regular CPU implementation, and arrive at an implementation that executes in the order of tenths of a second. The implemented multimodal perception module (including stereovision, binaural sensing and inertial sensing) builds an egocentric representation of occupancy and local motion, the Bayesian Volumetric Map (BVM), based on which gaze shift decisions are made to perform active exploration and reduce the entropy of the BVM. Experimental results show that the real-time implementation successfully drives the robotic system to explore areas of the environment mapped with high uncertainty.

Keywords GPU · NVIDIA CUDA · Multimodal Bayesian perception

J. F. Ferreira · J. Lobo · J. Dias (✉)
ISR-University of Coimbra, 3030-290 Coimbra, Portugal
e-mail: jorge@isr.uc.pt

J. F. Ferreira
e-mail: jfilipe@isr.uc.pt

J. Lobo
e-mail: jlobo@isr.uc.pt

1 Introduction

Perception has as of recently been regarded as a computational process of unconscious, probabilistic inference. Aided by developments in statistics and artificial intelligence, researchers have begun to apply the concepts of probability theory rigorously to problems in biological perception and action [22]. One striking observation from this work is the myriad ways in which human observers behave as near-optimal Bayesian observers. Several authors even argue that the brain codes even complex patterns of sensory uncertainty in its internal representations and computations—see for example [4, 10, 22].

In recent years, a variety of Bayesian inference algorithms for artificial perception inspired by studies of biological systems have been proposed. Their high computational cost has been a prohibitory factor for their actual implementation in real-time applications. In most cases, this computational cost results from probabilistic models requiring complicated inference techniques; however, in a smaller but extremely important subset of Bayesian inference algorithms, the problem lies, not with the inference mathematics per se, which can be rather simple, but with the large data structures involved, raising the issue of scalability.

In the meanwhile, graphics processing units (GPUs) have been progressively and rapidly advancing from being

specialised fixed-function to being highly programmable and incredibly powerful parallel computing devices. With the introduction of the Compute Unified Device Architecture (CUDA), GPUs are no longer exclusively programmed using graphics APIs. In CUDA, a GPU can be exposed to the programmer as a set of general-purpose shared-memory single instruction multiple data (SIMD) multicore processors. The number of threads that can be executed in parallel on such devices is currently in the order of hundreds and is expected to multiply soon. Many applications that are not yet able to achieve satisfactory performance on CPUs can get the benefit from the massive parallelism provided by GPUs [33, 34].

In this text, we will present a real-time implementation of an extense Bayesian framework for robotic multisensory perception (including stereovision, binaural sensing and inertial sensing) using CUDA. We will demonstrate that the SIMD features of GPUs provide a means of dealing with the scalability of highly parallelisable algorithms operating on large data structures, drastically improving overall processing rates comparing with CPU implementations, therefore allowing real-time performance. Moreover, we will demonstrate that CUDA is a very useful tool for GPU programming, as it provides a simplified yet powerful abstraction to graphic card intricacies [20].

2 Related work

GPUs have developed from fixed function architectures to programmable, multicore architectures, leading to new applications.

A relatively popular subset of this work over the years has been vision and imaging applications. Fung and Mann [17], present an excellent summary on this work, ranging from general purpose GPU (GPGPU) processing, where graphics hardware is used to perform computations for tasks other than graphics, to the more recent trend of *GPU Computing*, where GPU architectures and programming tools have been developed that have created a parallel programming environment that is no longer based on the graphics processing pipeline, but still exploits the parallel architecture of the GPU—in fact, GPU Computing has transformed the GPGPU concept into the simple mapping of parellelisable algorithms onto SIMD format for the GPU, making a complete abstraction from the intricacies of graphics programming.

As a result, several full-fledged computer vision and image processing toolkits and libraries that resort to GPU technology have emerged, such as OpenVIDIA [18], GPU4Vision [1] or GpuCV [12].

On the other hand, probabilistic approaches to perception have risen the stakes regarding the usefulness of GPU implementations of parallelisable algorithms. Neural network implementation is an example of this, as shown by Jang et al. [21], who propose a quick and efficient implementation of neural networks on both GPU and multicore CPU, with which they developed a text detection system, achieving performances about 15 times faster than the analogous implementation using CPU and about 4 times faster than implementation on GPU alone.

Occupancy grid-based sensor fusion algorithms, on the other hand, an example of a probabilistic approach to sensor fusion, have as of recently been a source of very interesting work on GPUs, given their obvious parallelisable trait due to the probability independence postulate between grid cells. Moreover, computational frameworks such as this are perfect candidates for GPU processing: very large data structures are processed in parallel using simple operations, yielding the perfect backdrop for SIMD-based computation. However, GPU implementations for such algorithms are still very recent and few—examples would be the work by Reinbothe et al. [37], and also Yguel et al. [39].

Hence we believe that there is a real contribution to be made in this area, specially now, when GPU Computing has taken such a huge step forward, with the appearance of tools such as NVIDIA's CUDA architecture [20], which will be summarised in the following section.

3 The Compute Unified Device Architecture (CUDA)

We will make a brief presentation of the main features of NVIDIA's CUDA, based on the excellent summary by Hussein et al. [19]. For a detailed description, refer to [31].

3.1 Hardware architecture

In CUDA terminology, the GPU is called the *device* and the CPU is called the *host*. A CUDA device consists of a set of multicore processors. Each multicore processor is simply referred to as a *multiprocessor*. Cores of a multiprocessor work in a SIMD fashion. All multiprocessors have access to three common memory spaces (globally referred to as *device memory*). They are:

Constant Memory: read-only cached memory space.
Texture Memory: read-only cached memory space that is optimised for texture fetching operations.
Global Memory: read/write non-cached memory

Besides the three memory spaces that are common among all multiprocessors, each multiprocessor has an on chip *shared memory* space that is common among its cores. Furthermore, each core has an exclusive access to a read/write non-cached memory space called *local memory*.

Accessing constant and texture memory spaces is as fast as accessing registers on cache hits. Accessing shared memory is as fast as accessing registers as long as there is no bank conflict. On the other hand, accessing global and local memory spaces is much slower, typically two orders of magnitude slower than floating point multiplication and addition.[1]

3.2 Execution model

The execution is based on *threads*. A thread can be viewed as a module, called a *kernel*, that processes a single data element of a data stream. Threads are batched in groups called *blocks*, and can only access shared memory from within their respective blocks. The group of blocks that executes a kernel constitutes one *grid*. Each thread has a three-dimensional index that is unique within its block. Each block in a grid in turn has a unique two-dimensional index. Knowing its own index and the index of the block in which it resides, each thread can compute the memory address of a data element to process.

A block of threads can be executed only on a single multiprocessor. However, a single multiprocessor can execute multiple blocks simultaneously by time slicing. Threads in a block can communicate with one another via the shared memory space. They can also use it to share data fetched from global memory. There is no means of synchronisation among threads in different blocks. The number of threads within a block that can execute simultaneously is limited by the number of cores in a multiprocessor. A group of threads that execute simultaneously is called a *warp*. Warps of a block are concurrently executed by time slicing.

3.3 Optimisation issues

There are some important considerations that need to be taken into account to obtain good performance on CUDA.

- Effect of Branching: If different threads of a warp take different paths of execution, the different paths are serialised, which reduces parallelism.
- Global Memory Read Coalescing: Global memory reads from different threads in a warp can be coalesced. To be coalesced, the threads have to access data elements in consecutive memory locations. Moreover, addresses of all data elements must follow the memory alignment guidelines. Details are in [31].
- Shared Memory Bank Conflict: Reading from shared memory is as fast as reading from registers unless a bank conflict occurs among threads. Simultaneous accesses to the same bank of shared memory are in most cases serialised.
- Writing to Global Memory: In CUDA, two or more different threads, in the same warp, can write simultaneously to the same address in global memory. The order of writing is not specified, but, one is guaranteed to succeed.

[1] However, the new Fermi GPUs from NVIDIA will have Configurable L1 and Unified L2 caches [32]. Refer to concluding section for further discussion on the subject.

4 Bayesian framework for active multimodal perception

4.1 Bayesian models for multimodal perception

Active perception has been an object of study in robotics for decades now, specially active vision, which was first introduced by Bajcsy [3] and later explored by Aloimonos et al. [2]. Many perceptual tasks tend to be simpler if the observer is active and controls its sensors [2]. Active perception is thus an intelligent data acquisition process driven by the measured, partially interpreted scene parameters and their errors from the scene. The active approach has the important advantage of making most ill-posed perception tasks tractable [2].

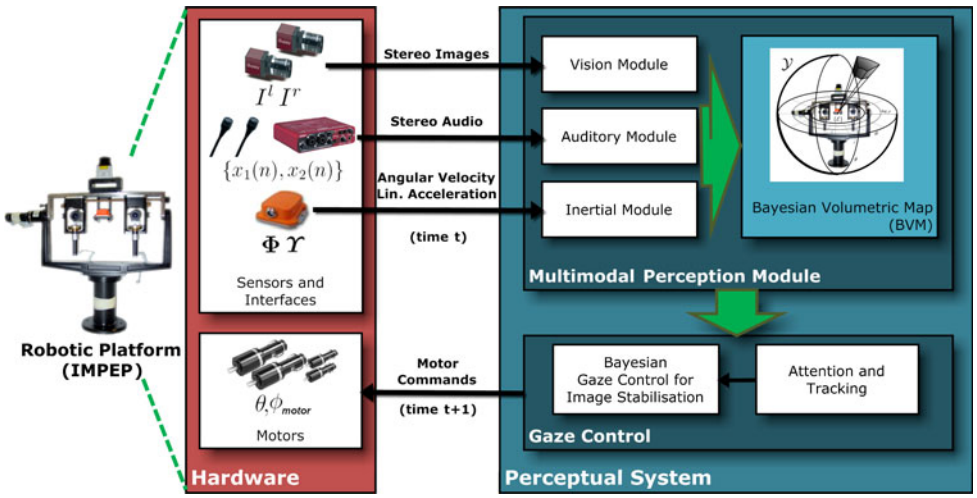
Active multisensory perception using spatial maps has, however, been the object of study since only much recently—an example of this research would be the work of Koene et al. [23] in visuoauditory-driven gaze shift generation.

The active multimodal perception system implemented in this work integrates a complete framework of Bayesian algorithms which jointly process the outputs yielded by visual, auditory and inertial sensors at a given instant in time, use the processed data to build a spatial representation of selected properties of the surrounding world, and compute a stabilised gaze shift towards a site on the environment to be explored in the subsequent time step—Fig. 1 shows an overview of the system's layout and integration.

4.1.1 Visuoauditory sensor fusion through Bayesian filtering using a log-spherical grid

A spatial representation framework for multimodal perception of 3D structure and motion, the Bayesian Volumetric Map (BVM), was presented in [13], characterised by an *egocentric*, *log-spherical spatial configuration* to which the Bayesian occupancy filter (BOF), as formalised by Tay et al. [38], has been adapted. It effectively provides a computational means of storing and updating a perceptual spatial map in a short-term working memory data-structure, representing both 3D structure and motion, without the need for any object segmentation process (see Fig. 2).

This model and its operators were developed using the *Bayesian Program* (BP) formalism, as first defined by



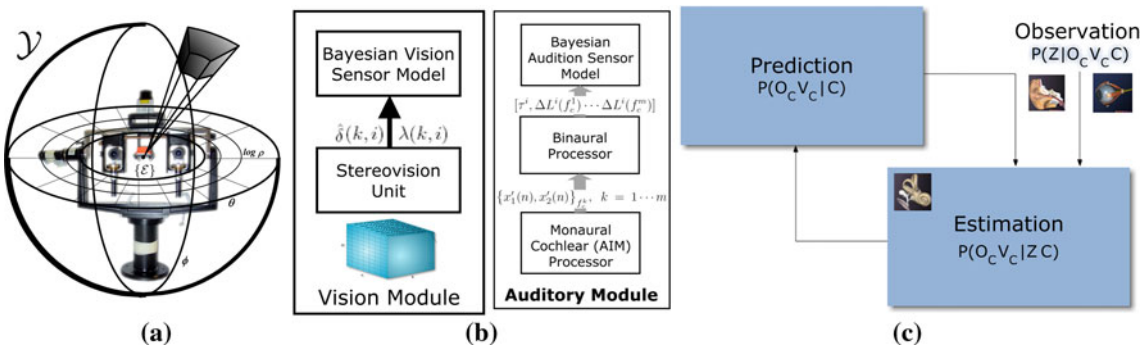


Fig. 1 Integration layout for the active multimodal perception system

Fig. 2 Multimodal perception framework details. **a** The Bayesian Volumetric Map (BVM) and the Integrated Multimodal Perception Experimental Platform (IMPEP); **b** BVM sensor models; **c** BVM Bayesian occupancy filter

Lebeltel [25] and later consolidated by Bessière et al. [5]. This formalism was created to supersede, restate and compare numerous classical probabilistic models such as Bayesian networks (BN), dynamic Bayesian networks (DBN), Bayesian filters, hidden Markov models (HMM), Kalman filters, particle filters, mixture models, or maximum entropy models. It also promotes hierarchical programming by making the relationship between models and submodels explicit, in a subroutine-like fashion.

In the BVM framework, cells of a partitioning grid on the BVM log-spherical space $\mathcal{Y}$ associated with the egocentric coordinate system $\{\mathcal{E}\}$ are indexed through $C \in \mathcal{C} \subset \mathcal{Y}$, where $\mathcal{C}$ represents the subset of positions in $\mathcal{Y}$ corresponding to the "far corners" ($\log_b \rho_{\max}$, $\theta_{\max}$, $\phi_{\max}$) of each cell C, O_C is a binary variable representing the state of occupancy of cell C (as in the commonly used occupancy grids—see [11]), and V_C is a finite vector of random variables that represent the state of all local motion possibilities used by the prediction step of the Bayesian filter associated with the BVM for cell C, assuming a constant velocity hypothesis, as depicted on Fig. 2. Sensor measurements (i.e. the result of visual and auditory processing) are denoted by Z—observations $P(Z \mid O_C\ V_C\ C)$ are given by the Bayesian sensor models[2] of Fig. 2, which yield results already integrated within the log-spherical configuration.

By restricting egomotion to rotations around the egocentric axes, vestibular sensing (see following subsection), together with the encoders of the motors of the robotic head (i.e. proprioception), will yield measurements of angular velocity and position which can then be easily used to manipulate the BVM, which is, by definition, in spherical coordinates [13]. In this case, the most effective solution for integration is to perform the equivalent index shift. This process is described by redefining $C : C \in \mathcal{Y}$ *indexes a cell in the BVM by its far corner, defined as* $C = (\log_b \rho_{\max}, \theta_{\max} + \theta_{\text{inertial}}, \phi_{\max} + \phi_{\text{inertial}}) \in \mathcal{C} \subset \mathcal{Y}$.

For Bayesian stereovision sensing, we have decided to use a data structure loosely based on the neuronal population activity patterns found in the perceptual brain to represent uncertainty in the form of probability distributions [36]. Thus, a spatially organised 2D grid may have each cell associated with a "population code" extending to additional dimensions, yielding a set of probability values

[2] These sensor models are, in fact, Bayesian subprograms of the BVM.

encoding a N-dimensional probability distribution function. This information, conveniently expressed in cyclopean coordinates (thus using the egocentric frame of reference $\{\mathcal{E}\}$), is consequently used as soft evidence by a Bayesian sensor model previously presented in Ferreira et al. [13] and Fereira et al. [15] (Fig. 2).

The Bayesian binaural system, which was fully described in Pinho et al. [35], and Ferreira et al. [16], is composed of three distinct and consecutive processors (Fig. 2): the monaural cochlear unit, which processes the pair of monaural signals $\{x_1, x_2\}$ coming from the binaural audio transducer system by simulating the human cochlea, so as to achieve a tonotopic representation (i.e. a frequency band decomposition) of the left and right audio streams; the binaural unit, which correlates these signals and consequently estimates the binaural cues and segments each sound-source; and, finally, the Bayesian 3D sound-source localisation unit, which applies a Bayesian sensor model so as to perform localisation of sound-sources in 3D space, again using the egocentric frame of reference $\{\mathcal{E}\}$.

The BVM is extendible in such a way that other properties, characterised by additional random variables and corresponding probabilities might be represented, other than the already implemented occupancy and local motion properties O_C and V_C, by augmenting the hierarchy of operators through Bayesian subprogramming [5, 25].

One such property that we propose to model uses the knowledge from the BVM to determine gaze shift fixation sites. More precisely, it elicits gaze shifts towards locations of high entropy/uncertainty based on the rationale conveyed by an additional variable that quantifies the uncertainty-based interest of a cell on the BVM, thus promoting entropy-based active exploration.

Therefore, we introduce a new random variable U_C, which takes the algorithm presented in Ferreira et al. [14] (see Fig. 3) and expresses it in a compact mathematical form:

$$U_C = \begin{cases} (1 - P([O_C = 1]|C)) \frac{\|\nabla H(C)\|}{\max \|\nabla H(C)\|} & C \in \mathcal{F}, \\ 0 & C \notin \mathcal{F}. \end{cases} \quad (1)$$

where $\mathcal{F} \subset \mathcal{C}$ represents the set of *frontier cells*[3] and

$$H(C) \equiv H(V_C, O_C) = -\sum_{O_C, V_C} P(O_C V_C|C) \log P(O_C V_C|C)$$

and

$$\|\nabla H(C)\| = \|[H(C) - H(C_{\rho-}), H(C) - H(C_{\theta-}), H(C) - H(C_{\phi-})]^T\|$$

represent the generic expressions of the *joint entropy* and *joint entropy gradient* of a cell C, respectively, as defined in Ferreira et al. [14]. This implies that $U_C \in [0, 1]$, being close to 1 when uncertainty is high and C is a frontier cell, and $U_C \to 0$ when uncertainty is low or C is not a frontier cell. In the current implementation, we use the maximum value for this variable for the decision on the gaze shift, as described in Ferreira et al. [14].

4.1.2 Modelling vestibular sensing using inertial sensors

To process the inertial data, we follow the Bayesian model proposed by Laurens and Droulez [24], adapted and presented in Lobo, Ferreira and Dias [28] to the use of inertial sensors. Instead of the vestibular system we have MEMS (microelectromechanical systems) gyros and accelerometers providing angular velocity and linear acceleration measurements. The aim here is to provide an estimate for the current angular position of the system, that mimics the human vestibular perception—see Fig. 4. To overcome the non-linearity of the motion equations and the high dimension space of possible distributions, particle filtering is used. The fact that some motions are more probable than others in human head motion is also replicated in the robotic version, limiting periods of sustained acceleration and also long duration rotations at constant velocity.

In this model the orientation of the system in space is encoded using a rotation matrix Θ. Angular velocity of the head is encoded using the yaw y, pitch p and roll r conventions, as illustrated in Fig. 4. The rotation update is given by

$$\Theta^{t+\delta t} = \Theta^t \cdot \boldsymbol{R}(\delta y, \delta p, \delta r) \quad (2)$$

where

$$\boldsymbol{R}(y,p,r) = \begin{bmatrix} c(y)\cdot c(p) & c(y)\cdot s(p).s(r) - s(y)\cdot c(r) & c(y)\cdot s(p)\cdot c(r) + s(y)\cdot s(r) \\ s(y)\cdot c(p) & c(y)\cdot c(r) + s(y)\cdot s(p)\cdot s(r) & -c(y)\cdot s(r) + s(y)\cdot s(p)\cdot c(r) \\ -s(p) & c(p)\cdot s(r) & c(p)\cdot c(r) \end{bmatrix}$$

where $c()$ and $s()$ are short form for cos() and sin().

The instantaneous angular velocity is defined as the vector:

[3] Set of cells, each belonging to a particular line-of-sight (θ_C, ϕ_C) in the BVM, just preceding the first occupied cell in that direction.

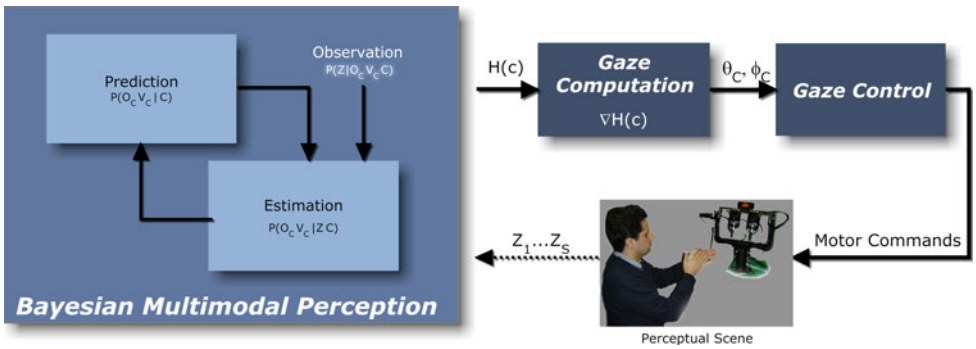


Fig. 3 Active multimodal perception using entropy-based exploration. Only the Bayesian models for multimodal perception and the entropy-based exploration algorithm implemented by the gaze computation module are described herewith; the gaze control module was presented in Lobo et al. [28]

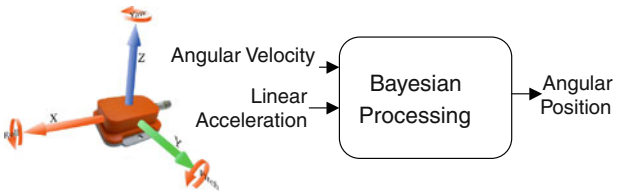


Fig. 4 Bayesian processing of inertial data

$$\Omega = \begin{pmatrix} \delta y/\delta t \\ \delta p/\delta t \\ \delta r/\delta t \end{pmatrix}$$

The gyros will measure Ω^t with added Gaussian noise, i.e. $\Phi^t = \Omega^t + \eta_\Phi^t$, where η_Φ^t is a three-dimensional vector, the elements of which follow independent Gaussian distributions with mean 0 and standard deviation σ_Φ.

The accelerometers will measure the gravito-inertial acceleration $\boldsymbol{F}$ with added Gaussian noise, i.e. $\Upsilon^t = \boldsymbol{F}^t + \eta_\Upsilon^t$, where η_Υ^t is a three-dimensional vector, the elements of which follow independent Gaussian distributions with mean 0 and standard deviation σ_Υ. $\boldsymbol{F}$ is the resultant acceleration due to linear acceleration and gravity. Given the geocentric body linear acceleration $\boldsymbol{A}$ and the system orientation Θ, we can compute $\boldsymbol{F}$. In a geocentric frame of reference gravity is a vector $\boldsymbol{G} = (0, 0, -9.81)$, and the gravito-inertial acceleration is given by $\boldsymbol{G} - \boldsymbol{A}$, transforming to the egocentric frame of reference we have

$$\boldsymbol{F} = \Theta^{-1} \cdot (\boldsymbol{G} - \boldsymbol{A}) \tag{3}$$

The state of our system at time t is therefore defined by $\xi^t = (\Theta^t, \Omega^t, \boldsymbol{A}^t, \boldsymbol{F}^t)$ and the sensor data by $\boldsymbol{S}^t = (\Phi^t, \Upsilon^t)$. In our case we only consider linear acceleration so that gravity can provide an absolute reference for orientation when $\boldsymbol{A} = 0$. Even in the absence of any sensory information, motion estimates for which the rotational velocity and acceleration are low are more probable. This can be described in a simple way using suitable Gaussian distributions for the priors $P(\boldsymbol{A}^t) \propto \mathcal{N}_{|\boldsymbol{A}^t|,0,\sigma_A}$ and $P(\Omega^t) \propto \mathcal{N}_{|\Omega^t|,0,\sigma_\Omega}$.

The Bayesian Program shown in Fig. 5 [28] is used to compute the probability distribution for the current state given all the previous sensory inputs and initial known distribution. At time t the Bayesian Program computes the probability distribution of the current state $\boldsymbol{\xi}^t$ given all the previous sensory inputs and the initial distribution. We can see also that the first-order Markov assumption is present in both the state dynamic model and sensor model: time dependence has a depth of one time step. The stationarity assumption is also implicit: models do not change with time. The filter iterates for each new time step, but the relationships between these variables remain the same for all time steps. This greatly reduces the complexity.

For the implementation the space of $\xi^{t-\delta t}$ that needs to be scanned has three dimensions: $\Theta^{t-\delta t}$. For a given $\xi^{t-\delta t}$, the space of possible ξ^t has three dimensions, so the total search space has six dimensions. Following the Bayesian

Program

Description

Specification

Relevant variables:

$\boldsymbol{\xi}^t = (\boldsymbol{\Theta}^t, \boldsymbol{\Omega}^t, \boldsymbol{A}^t, \boldsymbol{F}^t)$: state variables,

$\boldsymbol{S}^t = (\boldsymbol{\Phi}^t, \boldsymbol{\Upsilon}^t)$: sensor variables.

Decomposition:

$$P(\boldsymbol{\xi}^t \wedge \boldsymbol{\xi}^{t-\delta t} \wedge \boldsymbol{S}^t \ldots \wedge \boldsymbol{S}^0) = P(\boldsymbol{S}^t|\boldsymbol{\xi}^t) .P(\boldsymbol{\Omega}^t).P(\boldsymbol{A}^t).P(\boldsymbol{\Theta}^t|\boldsymbol{\Theta}^{t-\delta t} \wedge \boldsymbol{\Omega}^t) .P(\boldsymbol{\xi}^{t-\delta t} \wedge \boldsymbol{S}^{t-\delta t} \ldots \wedge \boldsymbol{S}^0)$$

Parametric forms:

$P(\boldsymbol{S}^t|\boldsymbol{\xi}^t) = P(\boldsymbol{\Phi}^t|\boldsymbol{\Omega}^t).P(\boldsymbol{\Upsilon}^t|\boldsymbol{F}^t)$:
sensor model, Gaussians and dirac;

$P(\boldsymbol{\Omega}^t)$, $P(\boldsymbol{A}^t)$:
a priori for state, Gaussians;

$P(\boldsymbol{\Theta}^t|\boldsymbol{\Theta}^{t-\delta t} \wedge \boldsymbol{\Omega}^t), P(\boldsymbol{F}^t|\boldsymbol{\Theta}^t \wedge \boldsymbol{A}^t)$:
state dynamic model, diracs;

$P(\boldsymbol{\xi}^{t-\delta t} \wedge \boldsymbol{S}^{t-\delta t} \ldots \wedge \boldsymbol{S}^0)$:
distribution computed on previous time step.

Identification:

Parameters of the Gaussians: σ_Φ, σ_Υ, σ_A and σ_Ω .

Question:

$P(\boldsymbol{\xi}^t|\boldsymbol{S}^t \wedge \ldots \wedge \boldsymbol{S}^0)$

Fig. 5 Bayesian program for processing of inertial data

model implementation proposed by Laurens and Droulez [24], we used a Particle Filter to perform the inference. A set of N particles, $\xi^{i,t}$, sample the state search space, and each one has an associated weight $w^{i,t} = P(\xi^{i,t})$. Starting from $\xi^{i,t-\delta t}$, we draw values for the Gaussians and apply Eqs. 2 and 3 to obtain $\xi^{i,t}$. The weighing factor is updated to $w^{i,t} = w^{i,t-\delta}t$. $P(\xi^{i,t})$. Resampling is applied, so that unlikely particles are deleted and likely ones are duplicated, in order to avoid having all particles drift towards improbable states. At each iteration a new set of N samples is drawn from the previous set of particles. Each particle of the previous set has a probability w^i to be chosen for each new particle. The weights in the new set are set to $1/N$.

An early approach to the problem implemented this program directly with aid of a C++ library, the ProBT inference engine [30]. For this work a CUDA implementation was implemented, allowing higher throughput to meet real-time requirements and more particles in the filter for better performance.

The particle filter is shown in Fig. 6. At each iteration a new set of N samples is drawn from the previous set. Each new particle is a copy of one of the previous ones, randomly chosen from the previous set when resampling. Each particle of the previous set has a probability $w^{i,t}$ to be chosen for each new particle. The random number generator was implemented using the algorithm described in [26]. Its output is then used with the Box-muller method [6], thus generating random numbers following a normal distribution.

4.2 The Integrated Multimodal Perception Experimental Platform (IMPEP)

To support our research work, an artificial multimodal perception system (IMPEP—Integrated Multimodal Perception Experimental Platform) has been constructed at the ISR/FCT-UC consisting of a stereovision, binaural and inertial measuring unit (IMU) setup mounted on a motorised head, with gaze control capabilities for image stabilisation and perceptual attention purposes—see Fig. 7.

The stereovision system is implemented using a pair of Guppy IEEE 1394 digital cameras from Allied Vision Technologies (http://www.alliedvisiontec.com), the binaural setup using two AKG Acoustics C417 linear microphones (http://www.akg.com/) and an FA-66 Firewire Audio Capture interface from Edirol (http://www.edirol.com/), and the miniature inertial sensor, Xsens MTi (http://www.xsens.com/), provides digital output of 3D acceleration, 3D rate of turn (rate gyro) and 3D earth-magnetic field data for the IMU. Initial pitch and roll positions are taken from the initial moment with the sensor at rest using the gravity acceleration.

Fig. 7 View of the current version of the Integrated Multimodal Perception Experimental Platform (IMPEP). The active perception head mounting hardware and motors were designed by the perception on purpose (POP—EC project number FP6-IST-2004-027268) team of the ISR/FCT-UC, and the sensor systems mounted at the Mobile Robotics Laboratory of the same institute, within the scope of the Bayesian approach to cognitive systems project (BACS−EC project number FP6-IST-027140)

The BVM–IMPEP framework was developed using the following software:

- *Vision sensor system*: With the OpenCV toolbox and David Gallup's implementation of a basic binocular stereo algorithm on GPU using CUDA (please refer to http://www.cs.unc.edu/~gallup/stereo-demo for more information). The algorithm reportedly runs at 40 Hz on 640 × 480 images at 50 disparities, computing left and right disparity maps and performing left-right consistency validation (which in our adaptation is used to produce the stereovision confidence maps).
- *Binaural sensor system*: Using an adaptation of the real-time software kindly made available by the Speech and Hearing Group at the University of Shefield [29] to implement binaural cue analysis as described in [35, 16].
- *Bayesian Volumetric Map, Bayesian sensor models and active exploration*: using our proprietary, parallel processing, single-precision GPU implementation

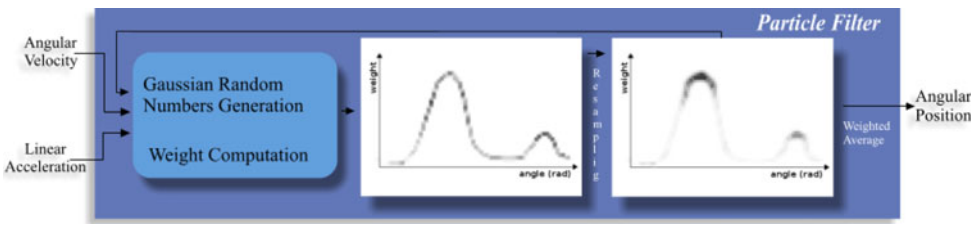


Fig. 6 Data flow during an iteration of the particle filter

developed with NVIDIA's CUDA, which is the main focus of this paper, described in Sect. 5.

The BVM–IMPEP system is composed of a local Ethernet network comprised of two PCs communicating and synchronising via Carmen messaging (Carmen Robot Navigation Toolkit—http://carmen.sourceforge.net/home.html—an open-source collection of software for mobile robot control sponsored by DARPA's MARS Program), one for all the sensory and BVM framework processing (including CUDA processing on a NVIDIA GeForce 9800 GTX, compute capability 1.1), and the other for controlling the IMPEP head motors, designed for portability (i.e. low-consumption and light-weight) in order to be mounted on mobile robotic platforms in the future—see Fig. 8. Both are equipped with Ubuntu Linux v9.04.

5 Implementation on GPU using CUDA

The activity diagram for the BVM Bayesian framework is presented in Fig. 9, depicting an inference step corresponding to time t and respective timeline. In the following lines, our GPU implementation of the BVM algorithms developed with NVIDIA's CUDA that exectute this timeline will be described in more detail.

5.1 Bayesian Volumetric Map filter

The BVM filter, which comprises the processing lane on the right of Fig. 9, launches kernels based on a single three-dimensional grid corresponding to the log-spherical configuration—see Fig. 10. In fact, both input matrices (i.e. observations and previous system state matrices) and output matrices (i.e. current state matrices) have the same indexing system. Blocks on this grid were arranged in such a way that their 2D indices would coincide with azimuth θ and elevation ϕ indices on the grid, assuming that the full N-depth of the log-distance index is always copied to shared memory.

By trial-and-error we arrived at the conclusion that block size was limited by shared memory resources to $5 \times 5 \times N$ for $N \leq 10$ and $3 \times 3 \times N$ for $N = 11$, which would therefore be the top limit for depth using this rationale of a single grid for the whole BVM space. In fact,

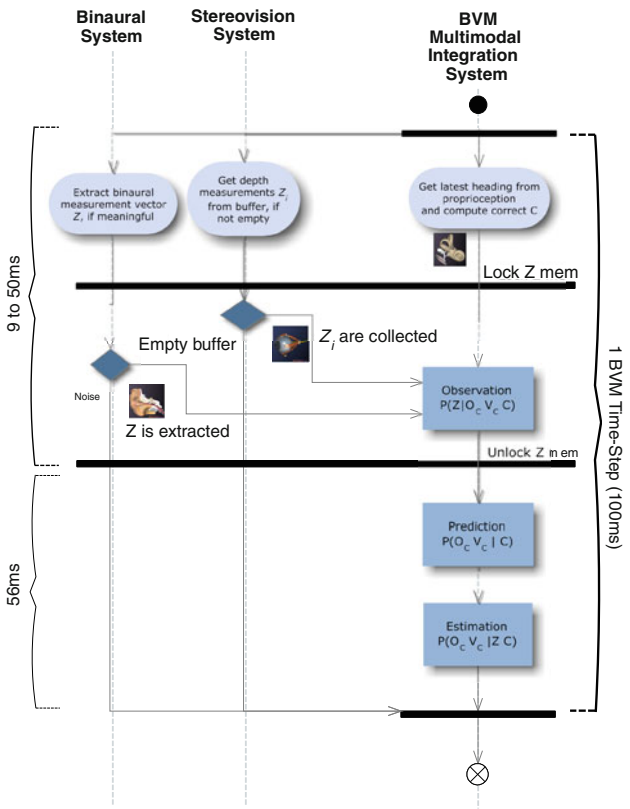


Fig. 9 Activity diagram for an inference time-step at time t. Each *vertical lane* represents a processing thread of the module labelled in the corresponding title. Maximum processing times (for $N = 10$, $\Delta\theta = 1°$, $\Delta\phi = 2°$) are also presented in the timeline for reference

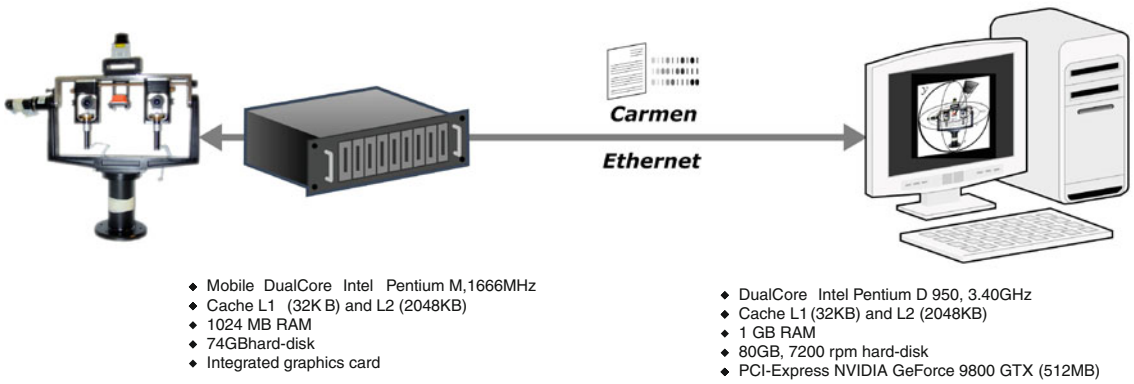


Fig. 8 BVM–IMPEP system network diagram

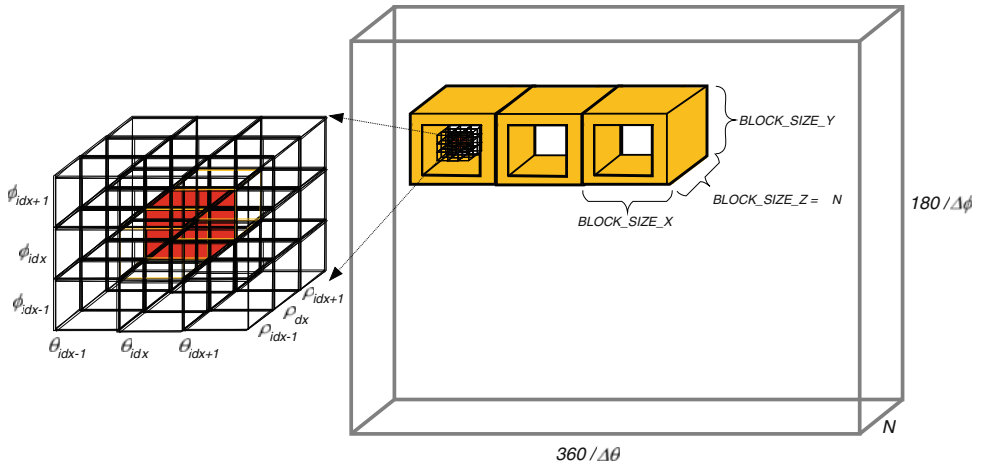


Fig. 10 BVM filter CUDA implementation. On the *right*, the overall 3D BVM grid is shown. On the *left*, a zoom in on the nine adjacent cells needed to update a central cell of the BVM are shown—this means that shared memory is required. As mentioned before, CUDA allows reference to each thread using a three-dimensional index; however, it only allows two-dimensional indexing for thread blocks. For this reason, we decided to assign the smallest dimension to the third axis (from now on referred to as "depth"—with size N), and by making all blocks the same depth as the global grid—this ensures that the block two-dimensional index corresponds to the remaining axes, simplifying memory indexing computations. Each thread loads its cell's previous state into shared memory and the log-probabilities for sensor measurements. The need for access to the previous states of adjacent cells further complicates the implementation by forcing the use of *aprons*, depicted in *yellow* within the thread blocks (see Fig. 12 for further details on kernel implementation using aprons)

for $N < 11$, there were 250 threads and 8,000 bytes of shared memory per block, thus limiting the maximum number of blocks per multiprocessor to 2 for the compute capability 1.1 of the GeForce 9800 GTX; for $N = 11$, on the other hand, there were 90 threads and 2,880 bytes of shared memory per block, increasing the limit of blocks per multiprocessor to 5.

The flowchart for the BVM filter kernel is shown on Fig. 12a.

5.2 Bayesian processing of stereovision, binaural and inertial data

The stereovision sensor model, which comprises the second processing lane from the left and the "Observation" box of Fig. 9, launches kernels based on two-dimensional grids corresponding to image configuration—see Fig. 11. In fact, its input matrices (left and right images, and disparity and confidence maps) have the same indexing system, while its output matrices (visual observation matrices) have the same indexing as the BVM grid of Fig. 10.

By trial-and-error we arrived at the conclusion that block size was limited by register memory resources to 16×16 for 640×480 images. This also ensured that it was a multiple of the warp size so as to achieve maximum efficiency.

The implementation of the binaural sensor model, corresponding to the processing lane on the left and the "Observation" box of Fig. 9, contrastingly, is very simple—a vector of binaural readings is used as an input and a grid as shown on Fig. 10, but without resorting to aprons (i.e. shared memory; see Figs. 10 and 12a for a detailed explanation of this notion), was used to update sensor model measurement data structures analogous to those of the stereovision sensor model, by referring to a lookup table with normal distribution parameters taken from the auditory system calibration procedure (refer to [35] for more information).

When there are visual and binaural measurements available simultaneously, two CUDA streams[4] are created (i.e. forked), one for each sensor model, and then destroyed (i.e. merged).

Finally, the particle filter for the processing of inertial data runs with 3,072 particles. There are 128 particles per each processing block and each particle is run by one thread only. The resampling process takes place in two stages, as depicted on Fig. 13. In the first stage each halfwarp of threads computes the particle with larger weight, that particle will be replicated for every halfwarp thread. Then the contiguous halfwarps exchange the particles with index 0 and 15. The second process is an inter-block resampling, where there are fixed routes. Particles in one block are always sent to a fixed memory space assigned to another block. This resampling process seems most adequate for the processor architecture and the CUDA framework.

[4] CUDA *streams* are concurrent lanes of execution that allow parallel execution of multiple kernels on the GPU.

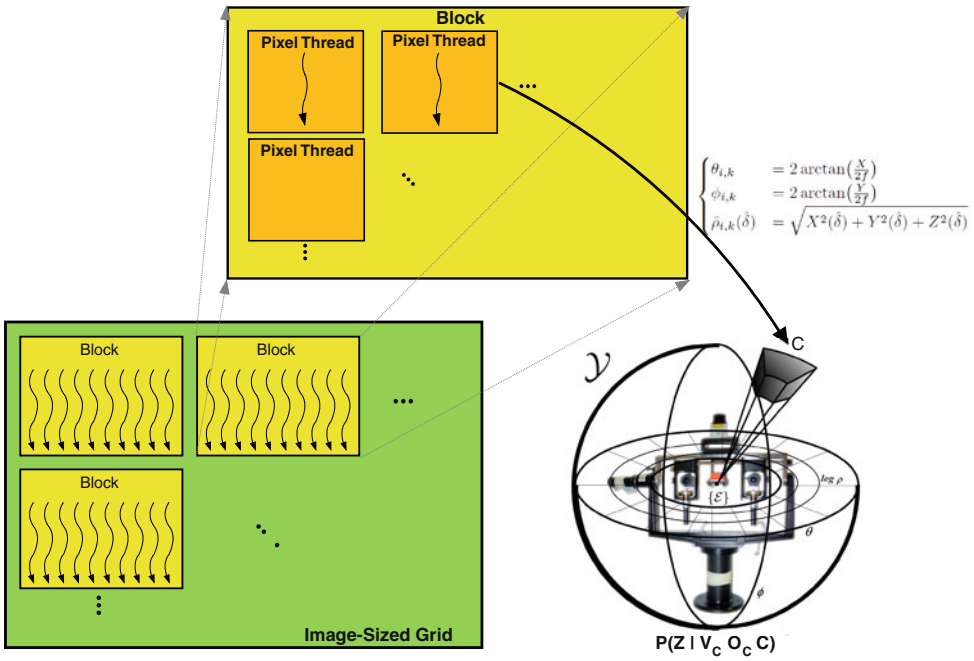


Fig. 11 Stereovision sensor model CUDA implementation. Each thread independently processes one pixel of the egocentric-referred depth map and confidence images (no use of shared memory required), computes the corresponding cell C on the BVM log-spherical spatial configuration using the equation shown, and updates two data structures in global device memory with that configuration storing log-probabilities corresponding to $P(Z \mid [O_C = 1]\ C)$ and $P(Z \mid [O_C = 0]\ C)$ (independent of velocity V_C), respectively. The update is performed using atomic summation operations provided by CUDA compute capability 1.1 and higher [31]. Atomic operations are needed due to the many-to-one correspondence between pixels and cells on the BVM; however, the order of summation is, obviously, non-important. Finally, since all atomic operations except "exchange" only accept integers as arguments, log-probabilities are converted from to floating-point to integer through a truncated multiplication by 10^n, with n corresponding to the desired precision (in our implementation, we used $n = 4$)

5.3 Active exploration

The active exploration algorithm was implemented resorting to CUDA atomic operations, global memory and four consecutive kernels in a sequential CUDA stream. This implementation is detailed on Fig. 12b.

6 Results

The real-time implementation of all the processes of the framework was subjected to performance testing for each individual module. Processing times and rates for the sensory systems are as follows:

- *Stereovision unit*: 15 Hz (using CPU), including image grabbing, preprocessing, stereovision processing itself (i.e. disparity and confidence map generation), and postprocessing and numerical conditioning.
- *Binaural processing unit*: Maximum rate of 40 Hz and 20 to 70 ms latency (using CPU) for 44 KHz, 16-bit audio, with 16 frequency channels and 50 ms buffer for cue computation.
- *Inertial processing unit*: 100 Hz using GPU. (The previous particle filter, running on a Pentium D CPU, achieved a maximum processing rate of 10 Hz, representing therefore a 10-times-faster performance.)

Processing times for the BVM modules are shown in Fig. 14. As can be seen, the full active exploration system runs from 6 to 10 Hz, depending on system parameters. This is ensured by forcing the main BVM thread to pause for each time-step when no visual measurement is available (e.g. during 40 ms for $N = 10$, $\Delta\phi = 2°$—see Fig. 9). This guarantees that BVM time-steps are regularly spaced, which is a very important requirement for correct implementation of prediction/dynamics, and also ensures that processing and memory resources are freed and unlocked regularly.

Running times for the Bayesian Volumetric Map update process decreased for each processing cycle from 5 to 30 min of serial processing on a Pentium Core 2 Quad CPU at 2.40 GHz, depending on BVM parameters, to a corresponding few hundredths of a second to a tenth of a second of parallel computing on an NVIDIA 9800 GTX

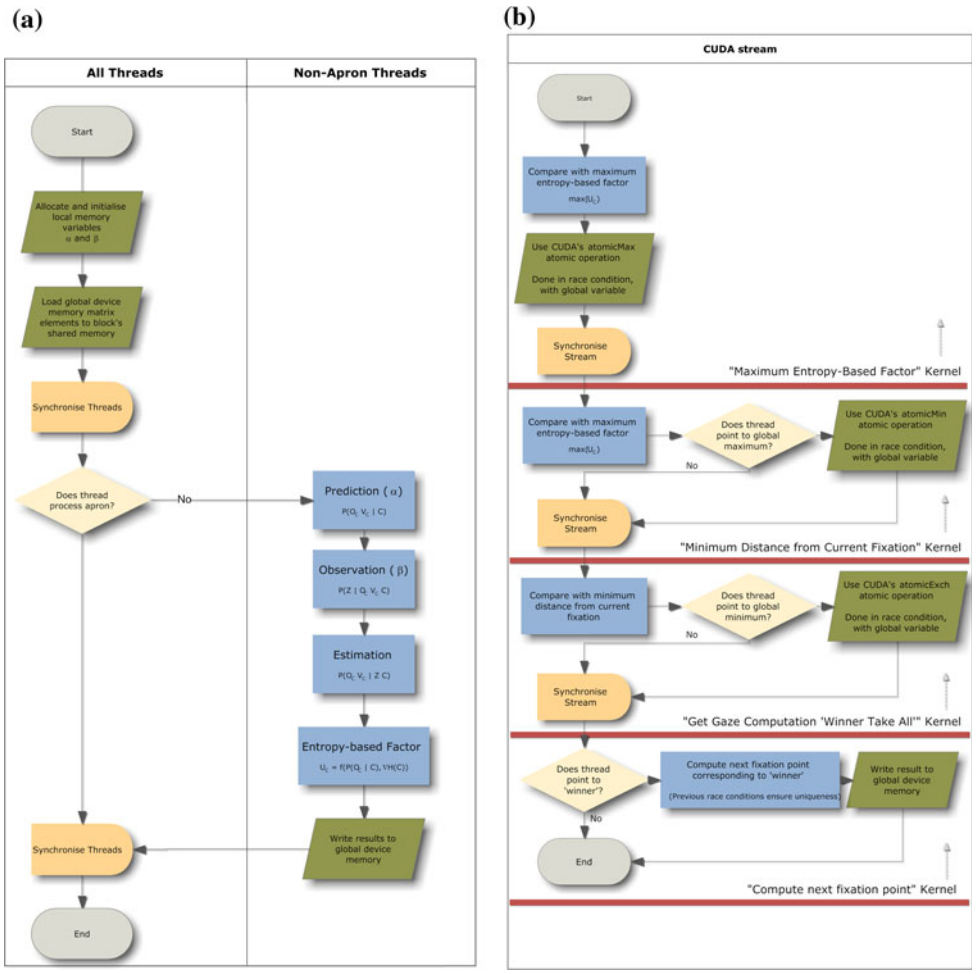


Fig. 12 BVM CUDA implementation flowcharts. **a** BVM filter CUDA kernel flowchart. *Aprons* are the limiting cells of the block, to which correspond threads that cannot access adjacent states, and therefore with the sole mission of loading their respective states into shared memory—thus, blocks must overlap as their indices change, so that all cells have the chance to be non-apron. After all threads, apron or non-apron, load their respective previous states into shared memory, all non-apron threads then perform Bayesian filter estimation and update the states, as depicted. The "Observation" box here denotes the computation of β by multiplying all available outputs from the stereovision and binaural Bayesian sensor models denoted as the "Observation" box of Fig. 9. **b** Active exploration CUDA stream flowchart. Four consecutive kernels in a sequential CUDA stream were used to implement the active exploration algorithm. The division of the processing workload into separate kernels was necessary due to the fact that the only way to enforce synchronisation between all concurrent CUDA threads in a *grid* (as opposed to all threads in a *block*, which are only a subset of the former) is to wait for all kernels running on that grid to exit—this is only possible at CUDA stream level (see main text for the definition of CUDA stream). CUDA atomic operations (refer to Fig. 11 for more information) and global memory were used to pass on data from one kernel to the next without the need for additional memory operations

graphics card, thus yielding an 18,000- to 30,000-times-faster performance.

Results for Bayesian processing of inertial data are presented in Fig. 15, showing the algorithm is robust enough to follow the attitude of the IMU based only on gyro and accelerometer information. The plot in red is the output of an extended Kalman filter from the IMU sensor firmware that also relies on magnetic sensor data to overcome gyro integration drift.

The results of processing a scenario testing different aspects of the full system are presented in Fig. 16. A scene consisting of two male speakers talking to each other in a cluttered lab is observed by the IMPEP active perception system and processed online by the BVM Bayesian filter, using entropy-based active exploration as described earlier, in order to scan the surrounding environment. These results show a projection of the log-spherical configuration onto Euclidean space of a volume approximately delimiting the so-called "personal space" (the zone immediately surrounding the observer's head, generally within arm's reach and slightly beyond, within 2 m range [9]) and the evolution of the exploration process through time.

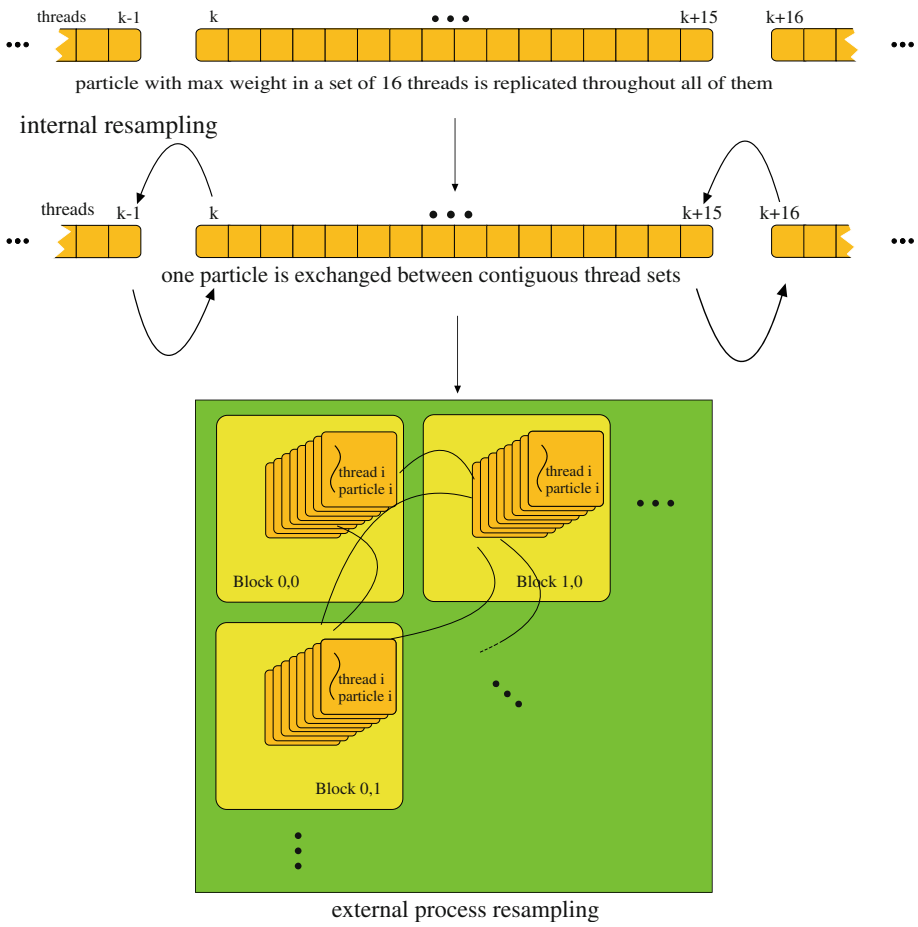


Fig. 13 CUDA implementation of the resampling process of the particle filter for the inertial model

7 Conclusions and future work

The active exploration algorithm thus successfully drives the IMPEP–BVM framework to explore areas of the environment mapped with high uncertainty in real-time, with an intelligent heuristic that minimises the effects of local minima by attending to the closest regions of high entropy first. Moreover, since the human saccade-generation system promotes fixation periods (i.e. time intervals between gaze shifts) of a few hundred milliseconds on average [7, 8], the overall rates of 6–10 Hz achieved with our CUDA implementation, in our opinion, back up the claim that our system does, in fact, achieve satisfactory real-time performance. In fact, running times for the Bayesian Volumetric Map update process decreased for each processing cycle from 5 to 30 min of serial processing on a CPU, depending on BVM parameters, to a corresponding few hundredths of a second to a tenth of a second of parallel computing on a GPU, thus yielding an 18,000–30,000-times-faster performance. The effect of double-precision processing used in some of the original baseline code to which current running times are compared to pales in face of the order of magnitude of performance increase.

Several improvements on the CUDA implementations described in this text are still possible, in order to increase the scalability of the system and improve processing times, namely memory coalescing through pitched 2D memory operations (refer to [31] for more information), possibly the use of pinned memory on the host, and the use of multiple grids processed by parallel CUDA streams for the BVM filter in order to subdivide the BVM data structure, therefore eradicating the limit of $N = 11$ divisions in distance. On the other hand, future use of the next generation of graphics cards and CUDA Compute architectures, such as the NVIDIA Fermi [32], will make a much improved computational framework and memory subsystem available, by adding, for example, more capacity, a hierarchy with Configurable L1 and Unified L2 Caches, ECC memory support and greatly improved atomic memory operation performance.

Additionally, in the future the vision module will be extended so as to include a Bayesian optical flow

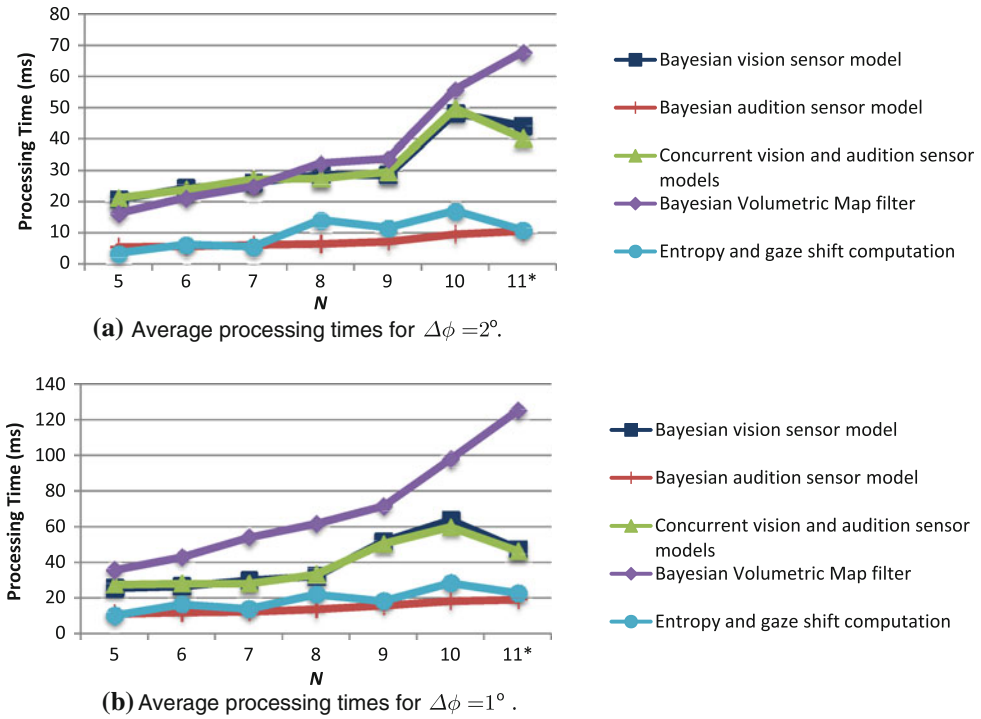


Fig. 14 BVM framework average processing times. Both *graphs* are for $\Delta\theta = 1°$, and show the average of processing times in ms for each activity depicted on Fig. 9, taken for a random set of 500 runs of each module in the processing of 5 dynamic real-world scenarios, with sensory horopter occupation varying roughly from 10 to 40% (although with no apparent effect on performance). These times are plotted against the number N of divisions in distance, which is the most crucial of system parameters (for $N > 11$, the GPU resources become depleted, and for $N < 5$ resolution arguably becomes unsatisfactory), and for two different reasonable resolutions in ϕ. Note that BVM filter performance degrades approximately exponentially with increasing resolution in distance, while the performance of all other activities degrades approximately linearly—the sole exception is the vision sensor model for $N = 11$, where it actually improves its performance. The reason for this is that the ratio of the effect of the influence of resolution on CUDA grid size versus the effect of the influence of resolution on the number of atomic operations required is reversed. (The * denotes that for $N = 11$ the block size is smaller for the BVM filter CUDA implementation—refer to main text for further details.)

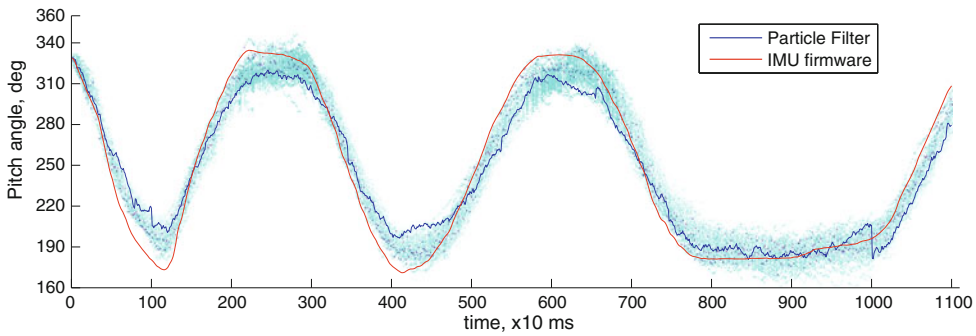


Fig. 15 Comparison between the outputs of the IMU firmware and particle filter processing of inertial data

processor, which will receive input in the form of a prior distribution built from the output from the inertial module. This extension, together with the stereovision unit, will allow automatic independent motion segmentation, as described in [27].

In summary, our artificial active perception system contributions are twofold:

– It provides a rather complete framework for active multimodal perception—introducing an approach which, while not strictly neuromimetic, finds its roots in the role of the dorsal perceptual pathway of the human brain—of which the proof of concept and relevance to robotic active perception is presented elsewhere [13–16, 35].

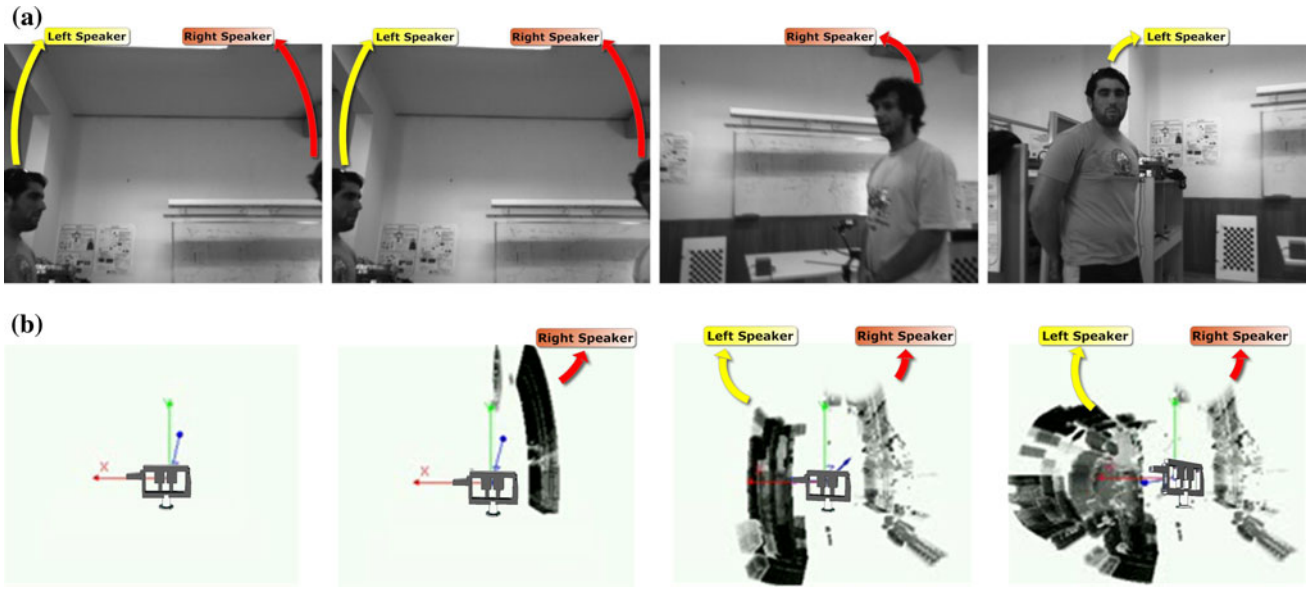


Fig. 16 Results for the real-time prototype for multimodal perception of 3D structure and motion using the BVM. A scene consisting of two male speakers talking to each other (left and right speakers both pinpointed for clarity) in a cluttered lab is observed by the IMPEP active perception system and processed online by the BVM Bayesian filter, using the active exploration heuristics described in the main text, in order to scan the surrounding environment. The parameters for the BVM are as follows: $N = 10$, $\rho_{\text{Min}} = 1{,}000$ mm and $\rho_{\text{Max}} = 2{,}500$ mm, $\theta \in [-180°, 180°]$, with $\Delta\theta = 1°$, and $\phi \in [-90°, 90°]$, with $\Delta\phi = 2°$, corresponding to $10 \times 360 \times 90 = 648{,}000$ cells, approximately delimiting the so-called "personal space" (the zone immediately surrounding the observer's head, generally within arm's reach and slightly beyond, within 2 m range [9]). **a** *Left camera snapshots* corresponding to chronologically ordered time-instants. Two male speakers are maintaining a dialog, at $-22°$ and 14° azimuth respectively relatively to the Z axis, which defines the frontal heading respective to the IMPEP "neck". As can be seen on the first frame, both speakers are initially outside the stereovision region-of-interest for processing, being consecutively scanned as a result of active exploration-driven gaze-shifts. **b** BVM results (frontal views, with Z pointing outward) corresponding to each of the snapshots in **a**. The *blue arrow* together with an oriented 3D sketch of the IMPEP perception system depicted in each map denote the current gaze orientation. Interpretation, from *left to right* (chronological evolution): (1) initial non-informative map; (2) sound coming from the speaker on the right triggers an estimate for occupancy from the binaural sensor model, and a consecutive exploratory gaze shift; (3) a few frames from the stereovision system trigger further evidence accumulation for occupancy by the vision sensor model at the gaze direction site, fusing readings from both sensory systems—higher spatial resolution from vision carves out the right speaker's silhouette from the first rough estimate from audition—while sound coming from the speaker on the left triggers an estimate for occupancy from the binaural sensor model, and a consecutive exploratory gaze shift in the speaker's direction; (4) after turning to new gaze direction site, the stereovision system triggers further evidence accumulation for occupancy at the left speaker's location, fusing readings from both sensory modalities—again, higher spatial resolution from vision carves out the left speaker's silhouette from the first rough estimate from audition

– By providing a real-time implementation of a probabilistic grid-based framework for multimodal perception, shown in Sect. 2 to be a relatively unexplored subject, it demonstrates that the SIMD features of GPUs provide a means of dealing with the scalability of highly parallelisable algorithms operating on large data structures, therefore allowing for real-time performances not possible using a CPU implementation.

Further details on ongoing work can be found at http://paloma.isr.uc.pt/~jfilipe/BayesianMultimodalPerception.

Acknowledgments The authors would like to thank José Marinho for his invaluable assistance on the implementation of the inertial sensor model particle filter. This publication has been supported by EC-contract number FP6-IST-027140, Action line: Cognitive Systems. The contents of this text reflect only the author's views. The European Community is not liable for any use that may be made of the information contained herein.

References

1. GPU4Vision—Accelerating Computer Vision. http://gpu4vision.icg.tugraz.at/ (2009)
2. Aloimonos, J., Weiss, I., Bandyopadhyay, A.: Active vision. Int. J. Comput. Vis. **1**, 333–356 (1987)
3. Bajcsy, R.: Active perception vs passive perception. In: Third IEEE Workshop on Computer Vision, Bellair, Michigan, pp 55–59 (1985)
4. Barber, M.J., Clark, J.W., Anderson, C.H.: Neural representation of probabilistic information. Neural Comput. **15**(8), 1843–1864 (2003). doi:10.1162/08997660360675062
5. Bessière, P., Laugier, C., Siegwart, R. (eds.): Probabilistic reasoning and decision making in sensory-motor systems. In: Springer Tracts in Advanced Robotics, vol 46. Springer. ISBN: 978-3-540-79006-8 (2008)

6. Box, G.E.P., Muller, M.: A note on the generation of random normal deviates. Ann. Math. Stat. **29**(2), 610–611 (1958)
7. Carpenter, R.H.S.: The saccadic system: a neurological microcosm. Adv. Clin. Neurosci. Rehabil. **4**, 6–8 (2004)
8. Caspi, A., Beutter, B.R., Eckstein, M.P.: The time course of visual information accrual guiding eye movement decisions. Proc. Natl. Acad. Sci. USA **101**(35), 13086–13090 (2004)
9. Cutting, J.E., Vishton, P.M.: Perceiving layout and knowing distances: the integration, relative potency, and contextual use of different information about depth. In: Epstein, W., Rogers, S. (eds.) Handbook of Perception and Cognition, vol 5; Perception of Space and Motion. Academic Press, New York (1995)
10. Denève, S., Latham, P.E., Pouget, A.: Reading population codes: a neural implementation of ideal observers. Nat. Neurosci. **2**(8), 740–745 (1999). doi:10.1038/11205
11. Elfes, A.: Using occupancy grids for mobile robot perception and navigation. IEEE Comput. **22**(6), 46–57 (1989)
12. Farrugia, J.P., Horain, P., Guehenneux, E., Alusse, Y.: GpuCV: a framework for image processing acceleration with graphics processors. In: 2006 IEEE International Conference on Multimedia and Expo, pp 585–588 (2006)
13. Ferreira, J.F., Bessière, P., Mekhnacha, K., Lobo, J., Dias, J., Laugier, C.: Bayesian models for multimodal perception of 3D structure and motion. In: International Conference on Cognitive Systems (CogSys 2008), University of Karlsruhe, Karlsruhe, Germany, pp 103–108 (2008)
14. Ferreira, J.F., Pinho, C., Dias, J.: Active exploration using Bayesian models for multimodal perception. In: Campilho, A., Kamel, M. (eds.) Image Analysis and Recognition. Lecture Notes in Computer Science Series (Springer LNCS), International Conference ICIAR 2008, pp 369–378 (2008)
15. Ferreira, J.F., Pinho, C., Dias, J.: Bayesian sensor model for egocentric stereovision. In: 14ª Conferência Portuguesa de Reconhecimento de Padrões Coimbra (RECPAD 2008) (2008)
16. Ferreira, J.F., Pinho, C., Dias, J.: Implementation and calibration of a Bayesian binaural system for 3D localisation. In: 2008 IEEE International Conference on Robotics and Biomimetics (ROBIO 2008), Bangkok, Thailand (2009)
17. Fung, J., Mann, S.: Using graphics devices in reverse: GPU-based image processing and computer vision. In: IEEE Int'l Conference on Multimedia and Expo, Hannover, Germany (2008)
18. Fung, J., Mann, S., Aimone, C.: OpenVIDIA: parallel GPU computer vision. In: ACM Multimedia 2005, Singapore, pp 849–852 (2005)
19. Hussein, M., Varshney, A., Davis, L.: On implementing graph cuts on CUDA. In: First Workshop on General Purpose Processing on Graphics Processing Units, Boston, MA (2007)
20. Hwu, W.M., Rodrigues, C., Ryoo, S., Stratton, J.: Compute unified device architecture application suitability. Comput. Sci. Eng. **11**(3), 16–26 (2009)
21. Jang, H., Park, A., Jung, K.: Neural network implementation using CUDA and OpenMP. In: Proceedings of the 2008 Digital Image Computing: Techniques and Applications, pp 155–161. IEEE Computer Society Washington, DC, USA (2008)
22. Knill, D.C., Pouget, A.: The Bayesian brain: the role of uncertainty in neural coding and computation. Trends Neurosci. **27**(12), 712–719 (2004)
23. Koene, A., Morén, J., Trifa, V., Cheng, G.: Gaze shift reflex in a humanoid active vision system. In: 5th International Conference on Computer Vision Systems (ICVS 2007), Applied Computer Science Group, Bielefeld University, Germany. ISBN:978-3-00-020933-8 (2007)
24. Laurens, J., Droulez, J.: Bayesian processing of vestibular information. Biol. Cybernet. **96**(4), 389–404 (2007)
25. Lebeltel, O.: Programmation Bayésienne des robots. PhD thesis, Institut National Polytechnique de Grenoble, Grenoble, France (1999)
26. L'Ecuyer, P.: Maximally equidistributed combined tausworthe generators. Math. Comput. **65**(213), 202–213 (1996)
27. Lobo, J., Ferreira, J.F., Dias, J.: Bioinspired visuovestibular artificial perception system for independent motion segmentation. In: Second International Cognitive Vision Workshop, ECCV 9th European Conference on Computer Vision, Graz, Austria. http://dib.joanneum.at/icvw2006/ (2006)
28. Lobo, J., Ferreira, J.F., Dias, J.: Robotic implementation of biological Bayesian models towards visuo-inertial image stabilization and gaze control. In: 2008 IEEE International Conference on Robotics and Biomimetics (ROBIO 2008), Bangkok, Tailand (2009)
29. Lu, Y.C., Christensen, H., Cooke, M.: Active binaural distance estimation for dynamic sources. In: Interspeech 2007, Antwerp, Belgium (2007)
30. Mekhnacha, K., Ahuactzin, J.M., Bessière, P., Mazer, E., Smail, L.: Exact and approximate inference in ProBT. Revue d'Intelligence Artificielle **21**(3), 295–332 (2007)
31. NVIDIA (2007) CUDA Programming Guide ver 1.2
32. NVIDIA (2009) NVIDIA's Next Generation CUDA™ Compute Architecture: Fermi™. Whitepaper, NVIDIA. http://www.nvidia.com/content/PDF/fermi_white_papers/NVIDIA_Fermi_Compute_Architecture_Whitepaper.pdf
33. Owens, J.D., Luebke, D., Govindaraju, N., Harris, M., Krger, J., Lefohn, A.E., Purcell, T.: A survey of general-purpose computation on graphics hardware. Comput. Graph. Forum **26**(5), 80–113 (2007)
34. Owens, J.D., Houston, M., Luebke, D., Green, S., Stone, J.E., Phillips, J.C.: GPU computing. In: Proceedings of the IEEE, vol 96, pp 879–899 (2008)
35. Pinho, C., Ferreira, J.F., Bessière, P., Dias, J.: A Bayesian binaural system for 3D sound-source localisation. In: International Conference on Cognitive Systems (CogSys 2008), pp 109–114. University of Karlsruhe, Karlsruhe, Germany (2008)
36. Pouget, A., Dayan, P., Zemel, R.: Information processing with population codes. Nat. Rev. Neurosci. **1**, 125–132 (2000)
37. Reinbothe, C., Boubekeur, T., Alexa, M.: Hybrid ambient occlusion. In: Proceedings of the Eurographics Symposium on Rendering (2009)
38. Tay, C., Mekhnacha, K., Chen, C., Yguel, M., Laugier, C.: An efficient formulation of the Bayesian occupation filter for target tracking in dynamic environments. Int. J. Veh. Auton. Syst. **6**, 155–171 (2008)
39. Yguel, M., Aycard, O., Laugier, C.: Efficient GPU-based construction of occupancy grids using several laser range-finders. Int. J. Veh. Auton. Syst. **6**(1–2), 48–83 (2007)

Author Biographies

João Filipe Ferreira completed his Electrical Engineering degree (specialisation in computers) from the Faculty of Sciences and Technology, University of Coimbra (FCTUC) in July 2000. He received the M.Sc. degree in Electrical Engineering from the same faculty, specialisation in Automation and Robotics, in January 2005. He is currently a PhD student at the FCTUC, working on the subject "Bayesian Cognitive Models for 3D Structure and Motion Multimodal Perception", sponsored by a PhD scholarship from the Portuguese Foundation for Technology and Sciences (FCT). He is a staff researcher at the Institute of Systems and Robotics (ISR), Coimbra Pole, since 2001, and a staff researcher for the FCTUC team on the European Integrated Project "Bayesian Approach to Cognitive Systems" (FP6-IST-27140), since 2006. His main research interests are human and artificial perception (mainly computer vision), robotics, image processing and 3D scanning. He conducts his

research on Bayesian cognitive models for multimodal artificial perception systems at the Electrical Engineering Department of the FCTUC, and research on human multimodal perception at the Biomedical Institute for Research in Light and Image (IBILI), Faculty of Medicine of the University of Coimbra.

Jorge Lobo completed his five year course in electrical engineering at Coimbra University in 1995. In April 2002, he received the MSc degree with the thesis "Inertial Sensor Data Integration in Computer Vision Systems." In June 2007, he received the PhD degree with the thesis "Integration of Vision and Inertial Sensing." He was a junior teacher in the Computer Science Department of the Coimbra Polytechnic School, and later joined the Electrical and Computer Engineering Department of the Faculty of Science and Technology at the University of Coimbra, where he currently works as Assistant Professor. His current research is carried out at the Institute of Systems and Robotics, University of Coimbra, and is aimed at the fusion of inertial information with vision systems in mobile robots.

Jorge Dias born on March 7, 1960, in Coimbra, Portugal and has a PhD degree on Electrical Engineering at University of Coimbra, specialization in Control and Instrumentation, November 1994. Jorge Dias conducts his research activities at the Institute of Systems and Robotics (ISR—Instituto de Sistemas e Robótica) at University of Coimbra. Jorge Dias' research area is Computer Vision and Robotics, with activities and contributions on the field since 1984. He has several publications on Scientific Reports, Conferences, Journals and Book Chapters. Jorge Dias teaches several engineering courses at the Electrical Engineering and Computer Science Department, Faculty of Science and Technology, University of Coimbra. He is responsible for courses on Computer Vision, Robotics, Industrial Automation, Microprocessors and Digital Systems. He is also responsible for the supervision of Master and PhD students on the field of Computer Vision and Robotics. Jorge Dias is currently director of LAS—Laboratory of Systems and Automation (former LAT), at the technology transfer institute IPN—Instituto Pedro Nunes. He was the main researcher of projects financed by the Portuguese Foundation for Science and Technology (FCT), by the European Community and by NATO Science for Stability Program. He is currently the vice-president of "Sociedade Portuguesa de Robótica" (SPR) and officer for the Portuguese Chapter for IEEE–RAS (Robotics and Automation Society).