3D Res. **2,** 04(2011)6
10.1007/3DRes.04(2011)6

3DR EXPRESS

Multi-sensor 3D Volumetric Reconstruction Using CUDA

Hadi Aliakbarpour • Luis Almeida • Paulo Menezes • Jorge Dias

Received: 11 August 2011 / Revised: 24 September 2011 / Accepted: 10 October 2011

Abstract This paper presents a full-body volumetric reconstruction of a person in a scene using a sensor network, where some of them can be mobile. The sensor network is comprised of couples of camera and inertial sensor (IS). Taking advantage of IS, the 3D reconstruction is performed using no planar ground assumption. Moreover, IS in each couple is used to define a virtual camera whose image plane is horizontal and aligned with the earth cardinal directions. The IS is furthermore used to define a set of inertial planes in the scene. The image plane of each virtual camera is projected onto this set of parallel-horizontal inertial-planes, using some adapted homography functions. A parallel processing architecture is proposed in order to perform human real-time volumetric reconstruction. The real-time characteristic is obtained by implementing the reconstruction algorithm on a graphics processing unit (GPU) using Compute Unified Device Architecture (CUDA). In order to show the effectiveness of the proposed algorithm, a variety of the gestures of a person acting in the scene is reconstructed and demonstrated. Some analyses have been carried out to measure the performance of the algorithm in terms of processing time. The proposed framework has potential to be used by different applications such as smart-room, human behavior analysis and 3D tele-conference.

Keywords multi-view camera, auto-stereoscopic visualization, dynamic scenes, 3D rendering quality assessment, visual servoing.

1. Introduction

Performing 3D volumetric reconstruction of people is one of the major research topics in the computer vision area. In this paper we present a volumetric 3D reconstruction method using no planar ground assumption. In order to observe the scene, a sensor network is employed. Each node of the network is comprised of a couple of Inertial Sensor (IS) and camera. In each couple, the IS is used to define a virtual camera whose plane is horizontal and its axes are aligned to the earth cardinal directions. Moreover, a set of inertial-planes, which are parallel to each other and horizontal, is defined in the scene for the purpose of 3D data registration. The image planes of virtual cameras are projected onto these inertial-planes using a geometric method through the concept of homography.

This work is an extension of our previous work[1]. In this paper, after presenting a comprehensive description of the framework, we have provided a parallel processing architecture for the algorithm which allows us to have a real-time implementation of the proposed approach. An effective implementation using GPU-CUDA has been carried out. 3D reconstruction of a person acting in a scene is provided while he is performing different gestures. To analyse the system's performance, task time measurement were executed while changing some system parameters and are available on this text.

There have been many works in the area of 3D reconstruction. Khan in [2] proposed a homographic framework for the fusion of multi-view silhouettes. A marker-less 3D human motion capturing approach is introduced in [3] using multiple views. Zhang in [4] introduced an algorithm for 3D projective reconstruction based on infinite homography. Homography-based mapping is used to implement a 3D reconstruction algorithm by Zhang and Hanson in [5]. Sorman et al. in [6] presented a multi-view reconstruction method based on volumetric graph-cuts. Lai and Yilmaz in [7] used images from uncalibrated cameras for performing projective reconstruction of buildings based on Shape From Silhouette (SFS) approach where buildings structure is used to compute vanishing points. Feldmann et al. [8] utilized the volumetric 3D reconstruction for the aim of online body motion tracking system. A multi-resolution volumetric 3D object reconstruction has been proposed by Guerchouche et al. in [9]. 3D reconstruction of an object using uncalibrated images taken by a single camera is proposed by Azevedo et al. in [10]. Lee et al. in [11] applied a 3D reconstruction method using photo consistency in

Hadi Aliakbarpour[1](✉) • Paulo Menezes[1](✉) • Jorge Dias[1](✉) • Luis Almeida[2](✉)
[1]Institute of Systems and Robotics, Polo II, University of Coimbra, Portugal.
[2]Institute Polytechnic of Tomar, Tomar, Portugal.
Email: {hadi,paulo,jorge}@isr.uc.pt, laa@ipt.pt

images taken from uncalibrated multiple cameras. A dynamic calibration and 3D reconstruction using homography transformation is proposed by Zhang and Li in [12]. In [13] SFS is combined with stereo imaging for the sake of 3D reconstruction by Lin. Michoud in [14] proposed a method to eliminate appearing ghost object in SFS-based 3D reconstructions. Aliakbarpour and Dias in [15] proposed a method to SFS-based 3D reconstruction. 3D reconstruction of a dynamic scene is investigated in [16] by Calibi. Franco in [17] used a Bayesian occupancy grid to represent the silhouette cues of objects.

In order to have a real-time processing time many researchers have already started to use GPU-based (GP-GPU and CUDA) parallelization of their algorithms. Joao Filipe et al. in [18] proposed a real-time implementation of Bayesian models for perception through multi-modal sensors by using CUDA. Almeida et al. implemented the stereo vision head vergence Using GPU-based cepstral filtering [19]. A GPU-based background segmentation algorithm is proposed in [20] by Griesser et al. Ziegler in [21] proposed a GPU data structure for graphic and vision. Real-time space carving using CUDA is investigated in [22] by Nitschke et al. In [23] CUDA is used to accelerate advanced MRI reconstructions. GPU-based method is used in [24] by Waizenegger for the purpose of high resoulution and real-time reconstruction using visual hulls. A GPU-based shape from silhouette (SFS) algorithm is implemented in [25] by Yous et al. An approach for volumetric visual hull reconstruction, using a voxel grid that focuses on the moving target object, is proposed by Knoblauch et al. [26]. A real-time 3D reconstruction system is presented in [27] by Ladikos et al. to achieve real-time performance. Yguel et al. in [28] implemented a GPU-based construction of occupancy grids using several laser range-finders.

As mentioned[29,30], recently the use of MEMS-IS has been continuously increasing and its price is decreasing. Nowadays one can see many smart phones equipped with this sensor as well as equipped with camera. There have been many authors showing the advantages of coupling an IS with camera for different purposes such as facilitating the camera network calibration process or increasing the robustness of the calibration result [29,30,31,32,33]. IS is error-prone in sensing the heading direction (rotation in vertical axis) but there have been many methods to overcome such a weakness as can be seen in the literature [29,34]. Therefore similar to what is mentioned[29], we do not enter in the area of justifying the benefits of such a coupling and we just assume that each camera in the network is coupled with an IS.

The rest of this paper is arranged as following: Three dimensional data registration using inertial-planes is investigated in Sec. 2 where the geometric relations among the virtual-planes are also explored. Moreover, the parallelizing architecture and its implementation using GPU-CUDA are also proposed in Sec 2. Sec. 3 presents some experiments where an acting person in the scene is reconstructed in different cases. Moreover some analysis related to the processing is provided in the same section. Eventually Sec. 4 is dedicated to the conclusion part.

2. Three dimensional data registration using inertial planes

A framework to register the 3D data of the scene is proposed and explained in this section. In this paper, we use the following convention for mathematical symbols: Vectors and matrices are all in bold, except for rotation matrices, camera calibration matrices and homography transforms which appear in normal capital. 3D points appear in capital bold and 2D points in small bold. Superscript in a variable indicates the reference frame in which the variable is expressed. For transformation matrices, subscript indicate the origin system and superscript means the destination system.

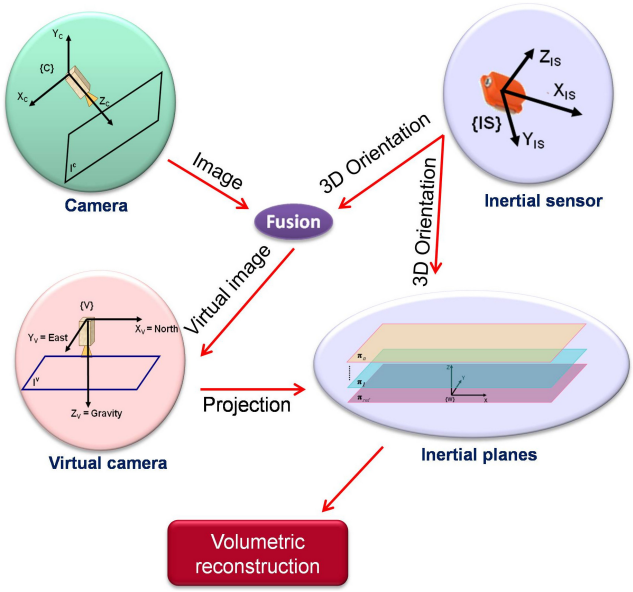

Fig. 1 Overall scheme of the proposed 3D volumetric reconstruction: 3D orientation from IS and image from camera are fused (using the concept of infinite homography) to define a downward-looking virtual camera whose axes are aligned to the earth cardinal direction (North-East-Down). 3D orientation from IS is as well as used to define a set of inertial-planes in the scene. The 3D reconstruction can be obtained by projecting the virtual images onto this set of parallel inertial planes.

2.1 Overall 3D reconstruction scheme

An overall scheme of the proposed volumetric reconstruction approach is depicted in Fig. 1. Two types of sensors are used: camera, for image grabbing and IS, for obtaining 3D orientation. Each camera is rigidly coupled to an IS. The outputs of each couple are fused using the concept of infinite homography and leads to have a downward-looking virtual camera whose axes are aligned to the earth cardinal direction (North-East-Down). Moreover, the 3D orientation of IS is used to define a set of inertial planes that are all virtual and parallel. The image planes of virtual cameras are projected onto this set of inertial-planes and the 3D volumetric reconstruction of the person (or generally an object) is obtained.

Regarding camera model, the pinhole camera model has been used [35]. In the pinhole camera model, a 3D point $\mathbf{X} = [X \quad Y \quad Z]^T$ from the scene is projected onto the

image plane of the camera as a 2D point, $\mathbf{x} = [x \quad y \quad 1]^T$, using the following model:

$$\mathbf{x} = K(R\mathbf{X} + \mathbf{t}) \tag{1}$$

where K is the camera calibration matrix, R and $\mathbf{t}$ are respectively rotation matrix and translation vector between the world and camera coordinate systems[35]. The camera matrix K, which is also referred as intrinsic parameter matrix, is defined by:

$$K = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

in which f_x and f_y represent the focal length of the camera in the directions of x and y. u_0 and v_0 are the elements of the principal point vector, P. In order to map points from one plane to another plane (with preserving the collinearity) the concept of homography [35] is used. Suppose a 3D plane is observed by two cameras. Moreover, assume that $\mathbf{x_1}$ and $\mathbf{x_2}$ are the image points of a 3D point $\mathbf{X}$ on the 3D plane. Then $\mathbf{x_1}$ and $\mathbf{x_2}$ are called a pair of corresponding points and the relation between them can be expressed as $\mathbf{x_2} = H\mathbf{x_1}$ in which H is a 3×3 matrix called planar homography induced by the 3D plane [36] and is equal to (up to scale)

$$H = K'(R + \frac{1}{d}\mathbf{t}\mathbf{n}^T)K^{-1} \tag{3}$$

where R and $\mathbf{t}$ are respectively rotation matrix and translation vector between the two cameras centers, $\mathbf{n}$ is Normal of the 3D plane, d is the orthogonal distance between the 3D plane and the camera center, eventually K and K' are intrinsic parameters of the two cameras (the first camera coordinate system is assumed as the world reference).

Fig. 2 shows a sensor network setup with a number of cameras. π_{ref} is an inertial plane, defined by the 3D orientation of IS, and is common for all cameras. Here $\{W\}$ is the world reference frame (a detailed specification of this reference frame shall be introduced in Sec. 2.2). In this setup, as mentioned before, each camera is rigidly coupled with an IS. The intention is to register a 3D point $\mathbf{X}$, observed by camera C, onto the reference plane $\boldsymbol{\pi}_{\mathbf{ref}}$ as ${}^{\pi}\mathbf{X}$ (2D), by the concept of homography and using inertial data. A virtual image plane is considered for each camera. Such a virtual image plane is defined (using inertial data) as a horizontal image plane at a distance f below the camera sensor, f being the focal length[37]. In other words, it can be thought that beside of each real camera C in the setup, a virtual camera V exists whose center, $\{V\}$, coincides to the center of the real camera $\{C\}$ (see Fig. 3). So that the transformation matrix between $\{C\}$ and $\{V\}$ just has a rotation part and the translation part is a zero vector.

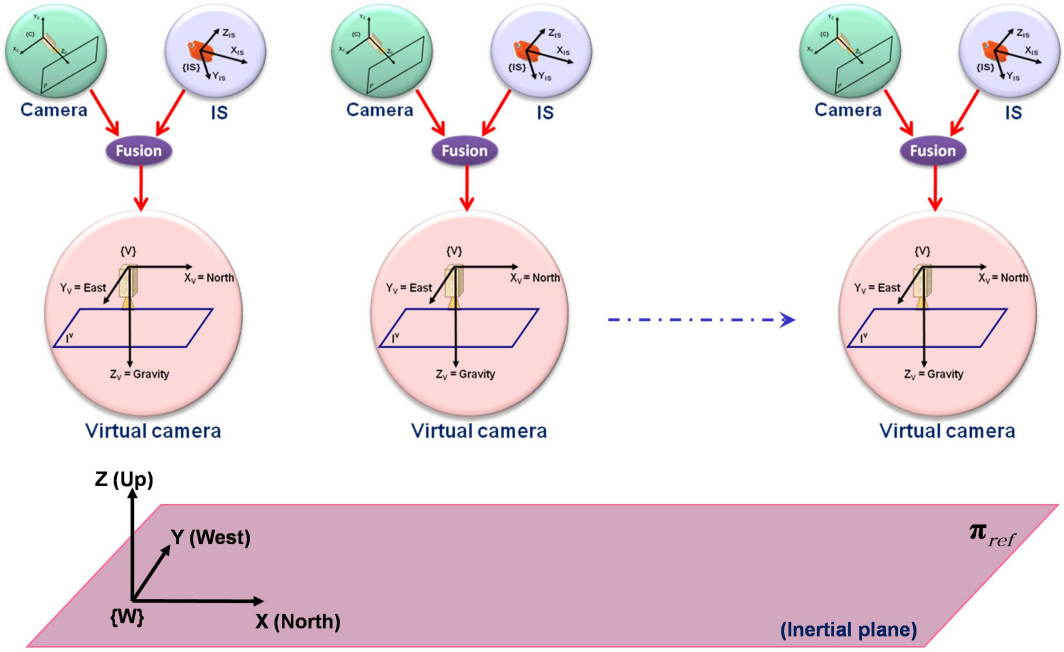

Fig. 2 A network of sensors observes a scene. The sensor network is comprised of a quantity of IS-camera couples. The inertial and visual information in each couple are fused using the concept of infinite homography which leads to define a virtual camera. π_{ref} is a virtual reference plane which is defined by using 3D orientation of IS and is common for all virtual cameras.

In order to register a 3D point $\mathbf{X}$ onto the π_{ref} as ${}^{\pi}\mathbf{X}$, three steps can be taken:

• First, the 3D point $\mathbf{X}$ is projected on the camera image plane by ${}^{C}\mathbf{x} = P\mathbf{X}$ (P is the projection matrix of the camera C).

• Second, ${}^{C}\mathbf{x}$ (the imaged point on the camera image plane) is projected to its corresponding point on the virtual camera's image plane as ${}^{V}\mathbf{x}$. Since this operation is plane to plane so it can be done by using ${}^{V}\mathbf{x} = {}^{V}H_{C}\,{}^{C}\mathbf{x}$) in which ${}^{V}H_{C}$ is a homography matrix[35].

• Third, the projected point on the virtual image plane, ${}^{v}\mathbf{x}$, is reprojected to the world virtual plane, π_{ref}, by having a suitable homography matrix, called ${}^{\pi}H_{v}$ (this operation is also plane to plane).

The first step it done by the camera based on the pinhole camera model (previously introduced). The second and third steps are described in the following two sub-sections. Assuming to already have ${}^{v}H_{c}$ and ${}^{\pi}H_{v}$, the final equation for registering a 3D point $\mathbf{X}$ onto the reference plane π_{ref} will be (see Fig. 5):

$$ {}^{\pi}\mathbf{x} = {}^{\pi}H_{v}\,{}^{v}H_{c}\,K\,(R\,\mathbf{X}+\mathbf{t}) \tag{6}$$

The way of obtaining ${}^{v}H_{c}$ (homography matrix between the real camera image plane and virtual camera image plane) and ${}^{\pi}H_{v}$ (homography matrix between the virtual camera image plane and the world 3D plane π_{ref}) is discussed in the next sub-sections by starting to describe the conventional coordinate systems.

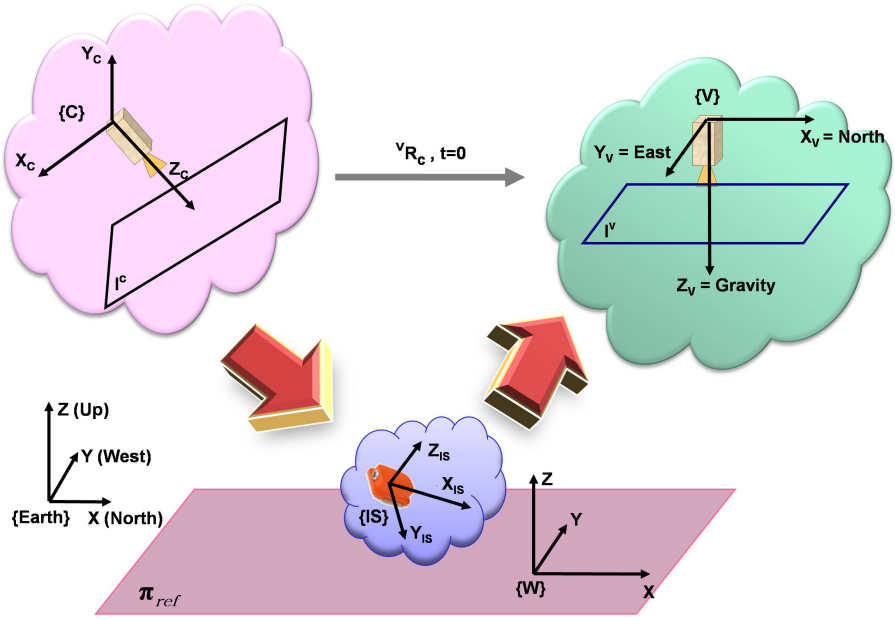

Fig. 3 Involved coordinate references in the definition of virtual camera; $\{Earth\}$: Earth cardinal coordinate system, $\{IS\}$: Inertial reference frame expressed in $\{Earth\}$, $\{W\}$: world reference frame of the framework, $\{C\}$: camera reference frame, $\{V\}$: reference frame of the virtual camera corresponding to $\{C\}$.

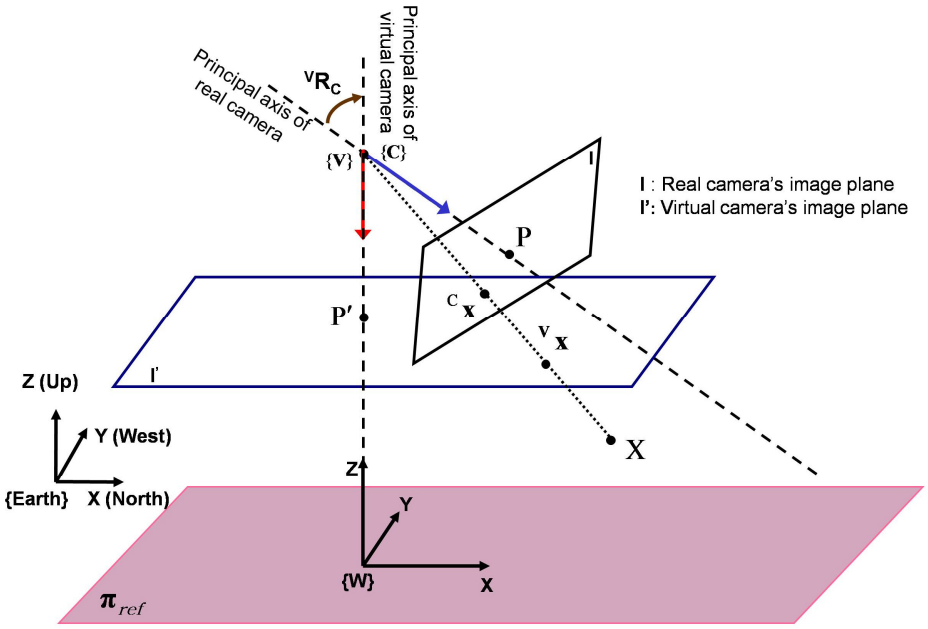

Fig. 4 Geometrical view of the virtual camera: The concept of infinite homography is used to fuse inertial-visual information and define an earth cardinal aligned virtual camera. Moreover using the inertial information, π_{ref} is defined as a virtual world plane which is horizontal and parallel to the image plane of virtual camera.

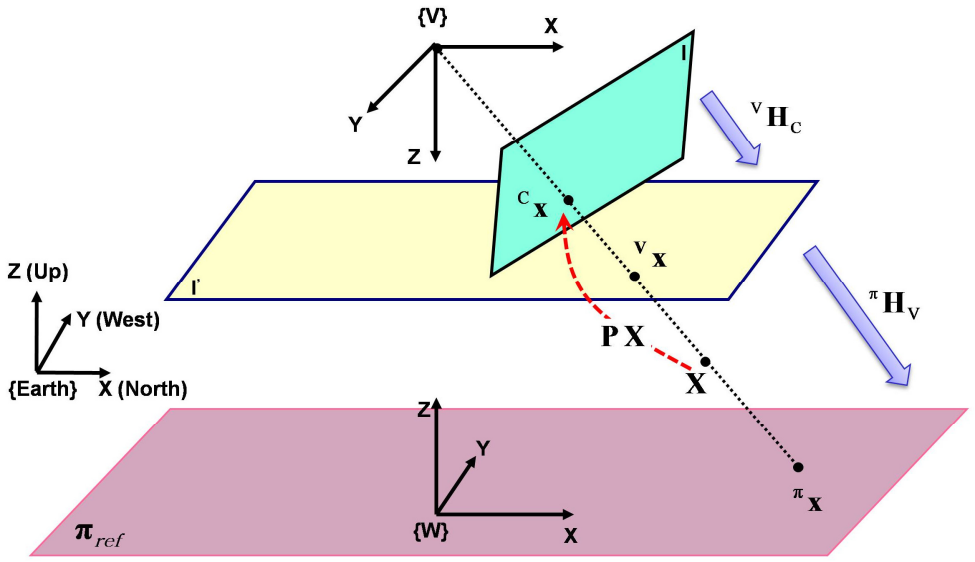

Fig. 5 One projection and two consecutive homographies are needed to register a 3D point $\mathbf{X}$ from the scene onto a world virtual plane π_{ref} through using IS. $^{V}H_{C}$: Homography from real camera image plane to the virtual one, $^{\pi}H_{V}$: Homography from the image plane of virtual camera to the reference inertial-plane π_{ref}.

2.2 Image plane of virtual camera

The definition of virtual camera is introduced in this sub-section. We start by presenting the coordinate systems. As seen in Fig. 3, there are five coordinate systems involved in this approach to be explained here:

- Real camera reference frame $\{C\}$: The local coordinate system of a camera C is expressed as $\{C\}$.
- Earth reference frame $\{E\}$: Which is an earth fixed reference frame having its X axis in the direction of North, Y in the direction of West and Z upward.
- Inertial Measuring Unit reference frame $\{IS\}$: This is the local reference frame of the IS sensor which is defined w.r.t. to the earth reference frame $\{E\}$.
- Virtual camera reference frame $\{V\}$: As explained, for each real camera C, a virtual camera V, is considered by the aid of a rigidly coupled IS to that. $\{V\}$ indicates the reference frame of such a virtual camera. The centers of $\{C\}$ and $\{V\}$ coincide and therefore there is just a rotation between these two references.

The idea is to use the 3D orientation provided by IS to register image data on the reference plane π_{ref} defined in $\{W\}$ (the world reference frame of this approach). The reference 3D plane π_{ref} is defined such a way that it spans the X and Y axis of $\{W\}$ and it has a normal parallel to the Z (See Fig. 3). In this proposed method the idea is to not using any real 3D plane inside the scene for estimating homography. Hence we assume there is no a real 3D plane available in the scene so our $\{W\}$ becomes a virtual reference frame and consequently π_{ref} is a horizontal virtual plane on the fly. Although $\{W\}$ is a virtual reference frame however it needs to be somehow specified and fixed in the 3D space. Therefore here we start to define $\{W\}$ and as a result π_{ref}. With no loss of generality we place $\mathbf{0}_{W}$, the center of $\{W\}$, in the 3D space such a way that $\mathbf{0}_{W}$ has a height d w.r.t the first virtual camera, V_0. Again with no loss of generality we specify its orientation the same as $\{E\}$ (earth fixed reference). Then as a result we can describe the reference frame of a virtual camera $\{V\}$ w.r.t $\{W\}$ via the following homogeneous transformation matrix

$$^{W}T_{V} = \begin{bmatrix} ^{W}R_{V} & \mathbf{t} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \tag{5}$$

where $^{W}R_{V}$ is a rotation matrix defined as (see Fig. 3):

$$^{W}R_{V} = \begin{bmatrix} \hat{\mathbf{i}} & -\hat{\mathbf{j}} & -\hat{\mathbf{k}} \end{bmatrix} \tag{6}$$

$\hat{\mathbf{i}}$, $\hat{\mathbf{j}}$ and $\hat{\mathbf{k}}$ being the unit vectors of the X, Y and Z axis, respectively, and $\mathbf{t}$ is a translation vector of the V 's center w.r.t $\{W\}$. Obviously using the preceding definitions and conventions, for the first virtual camera we have $\mathbf{t} = [0 \quad 0 \quad d]^{T}$.

Here we continue the discussion to obtain a 3×3 homography matrix $^{V}H_{C}$ which transforms a point $^{C}\mathbf{x}$ on the real camera image plane l to the point $^{V}\mathbf{x}$ on the virtual camera image plane l' as $^{V}\mathbf{x} = {}^{V}H_{C}\,{}^{C}\mathbf{x}$ (see Fig. 4). As described, the real camera C and virtual camera V have their centers coincided to each other, so the transformation between these two cameras can be expressed just by a rotation matrix. In this case $^{V}H_{C}$ is called infinite homography since there is just a pure rotation between real camera and virtual camera centers [35,38]. Such an infinite

homography can be obtained using a limiting process on Eq. (3) by considering $d \to \infty$ (as described in [35,36,39]):

$$ {}^{V}H_{C} = \lim_{d\to\infty} K\left({}^{V}R_{C} + \frac{1}{d}\mathbf{t}\mathbf{n}^{T}\right)K^{-1} = K\,{}^{V}R_{C}\,K^{-1} \quad (7) $$

where K is the camera matrix ${}^{V}R_{C}$ is the rotation matrix between $\{C\}$ and $\{V\}$. ${}^{V}R_{C}$ can be obtained through three consecutive rotations which is mentioned in Eq. (8) (see the reference frames in Fig. 3). The first one is to transform from real camera reference $\{C\}$ to the IS local coordinate $\{IS\}$, the second one transforms from the $\{IS\}$ to the earth fixed reference $\{E\}$ and the last one is to transform from $\{E\}$ to virtual camera reference frame $\{V\}$:

$$ {}^{V}R_{C} = {}^{V}R_{E}\;{}^{E}R_{IS}\;{}^{IS}R_{C} \quad (8) $$

${}^{IS}R_{C}$ can be obtained through a IS-camera calibration procedure. Here, Camera Inertial Calibration Toolbox [44] is used in order to calibrate a rigid couple of a IS and camera. Rotation from IS to earth, or ${}^{E}R_{IS}$, is given by the IS sensor w.r.t $\{E\}$. Since the $\{E\}$ has the Z upward but the virtual camera is defined to be downward-looking (with a downward Z) then the following rotation is applied to reach to the virtual camera reference frame:

$$ {}^{V}R_{E} = \left[\hat{\mathbf{i}} \quad -\hat{\mathbf{j}} \quad -\hat{\mathbf{k}}\right] \quad (9) $$

2.3 Projection of 3D data onto a world inertial plane

In this section we describe a method to find a homography matrix that transforms points from a virtual image plane I' (the image plane of virtual camera V) to the common world 3D plane $\boldsymbol{\pi}_{\mathbf{ref}}$ (recalling that these two planes are defined to be parallel. See Fig. 5). A 3D point $\mathbf{X}$ on π_{ref} is expressed in $\{W\}$ as $\mathbf{X} = [X \quad Y \quad 0 \quad 1]^{T}$ in its homogeneous form (recalling that XY-plane of $\{W\}$ corresponds to π_{ref} and therefore any points on this plane has $Z = 0$). For a general case (pinhole camera), $\mathbf{X}$ is projected on the image plane as following:

$$ \mathbf{x} = K[\mathbf{r}1 \quad \mathbf{r}2 \quad \mathbf{r}3 \quad \mathbf{t}]\begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = K[\mathbf{r}1 \quad \mathbf{r}2 \quad \mathbf{t}]\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \quad (10) $$

where $\mathbf{r}1$, $\mathbf{r}2$ and $\mathbf{r}3$ are the columns of the 3×3 rotation matrix, K is the camera calibration matrix (defined in Eq. 2) and $\mathbf{t}$ is the translation vector between $\boldsymbol{\pi}_{\mathbf{ref}}$ and camera center [35]. As can be seen Eq. (10) indicates a plane to plane projective transformation and therefore can be expressed like a planar homography:

$$ \mathbf{x} = {}^{V}H_{\pi}\,{}^{\pi}\mathbf{x} \quad (11) $$

where

$$ {}^{V}H_{\pi} = K[\mathbf{r}1 \quad \mathbf{r}2 \quad \mathbf{t}] \quad (12) $$

, ${}^{\pi}H_{V}$ denoting a 3×3 homography matrix and ${}^{\pi}\mathbf{x} = [X \quad Y \quad 1]^{T}$. Here we recall that for each camera within the network a virtual camera is defined (using inertial data). All such virtual cameras have the same rotation w.r.t world reference frame $\{W\}$. In other words it can be thought there is no rotation among the virtual cameras. ${}^{W}R_{V}$ or the rotation matrix between a virtual camera and $\{W\}$ was described through Eq. (6). Then considering ${}^{W}R_{V}$ from Eq. (6) and $\mathbf{t} = [t_1 \quad t_2 \quad t_3]^{T}$ as the translation vector, Eq. (12) can be formulated as :

$$ {}^{\pi}H_{V}^{-1} = K[\hat{\mathbf{i}} \quad -\hat{\mathbf{j}} \quad \mathbf{t}] = \begin{bmatrix} f_x & 0 & f_x t_1 + u_0 t_3 \\ 0 & -f_y & f_y t_2 + v_0 t_3 \\ 0 & 0 & t_3 \end{bmatrix} \quad (13) $$

It should be mentioned that the translation vector t can be obtained from different approaches. A method to estimate the translation vector among two cameras using two 3D points is proposed in our previous work[30]. In case of possibility of using GPS sensor (e.g. in outdoor scenarios), the translation can also be obtained from this device.

2.4 Extension for homographies among virtual camera and inertial-planes

In the preceding section the homography matrix from the image plane of a virtual camera V to the world 3D plane $\boldsymbol{\pi}_{\mathbf{ref}}$ was obtained as ${}^{\pi}H_{V}$ (see Eq. (13)). It is also desired to obtain the homography matrix from a virtual image to another world 3D plane parallel to $\boldsymbol{\pi}_{\mathbf{ref}}$ once we already have ${}^{\pi}H_{V}$. Lets consider $\boldsymbol{\pi}'$ as a 3D plane which is parallel to $\boldsymbol{\pi}_{\mathbf{ref}}$ and has a height Δh w.r.t it (see Fig. 5). ${}^{\pi'}H_{V}$ denotes the homography transformation which maps the points of the image plane of V onto $\boldsymbol{\pi}'$. By substituting t_3 in the equation (13) with $t_3 + \Delta h$, ${}^{\pi'}H_{V}$ can be expressed as a function of ${}^{\pi}H_{V}$ and Δh as follows:

$$ {}^{\pi'}H_{V}^{-1}(\Delta h) = {}^{\pi}H_{V}^{-1} + \Delta h P\hat{\mathbf{k}}^{T} \quad (14) $$

where $P = [u_0 \quad v_0 \quad 1]^{T}$ is the principal point of the camera V and $\hat{\mathbf{k}}$ is the unit vector of the Z axis.

2.5 3D Reconstruction using GPU-based parallel processing

Normally the volumetric reconstruction of person is time consuming due to the huge number data to be processed. In order to have a real-time processing (which is necessary for many applications) we propose a parallelizing of the 3D reconstruction algorithm.

GPU Hardware Architecture: in CUDA terminology, the GPU is called the device and the CPU is called the host (see Fig. 11). A CUDA device consists of a set of multi-core

processors. Each multicore processor is simply referred to as a multiprocessor. Cores of a multiprocessor work in a single instruction, multiple data (SIMD) fashion. All multiprocessors have access to three common memory spaces (globally referred to as device memory but with different access time). The CUDA program is organized into a host program, consisting of one sequential thread running on the host CPU, and several parallel kernels executed on the parallel processing device (GPU). A kernel executes a scalar sequential program on a set of parallel threads. The program organizes these threads into a grid of thread blocks.

The geometric models for projecting 3D data onto a set of virtual horizontal planes based on the concept of homography were previously introduced. Indeed here the homography transformation can be basically interpreted as shadow on each inertial-based virtual plane created by a light source located at the camera position. Considering several cameras (remembering light sources) which are observing the object then different shadows will appear on the inertial planes. For each inertial-plane, the intersection of all shadows on that gives the cross-section of that particular inertial-plane with the object. This interpretation is illustrated in the Fig. 6. The left figure shows an exemplary scene where a person is being observed by three cameras. In this figure an inertial-plane π_k is considered. The right figure shows a top view of the same scene which the shadows created by the three cameras. As can be seen the cross-section of the person with the inertial plane is obtained by intersecting all three shadows (the contour shown in white color). By considering a set of inertial-planes in different heights, obtaining the cross-section of each one with the object and stacking them over together the 3D volumetric reconstruction of the object will be obtained.

The proposed reconstruction approach is encapsulated and described as an algorithm in Alg. 7. First, the image plane of each virtual camera is obtained. Then the image of each virtual camera is projected onto a set of inertial-planes. N_c and N_π indicate the number of cameras and number of inertial-planes, respectively. I_{c_i} and I_{V_i} respectively are the image plane of camera C_i and its corresponding virtual camera V_i. Δh is the Euclidean distance among the inertial-planes which also can be interpreted as the vertical resolution of the algorithm. The labels 'Gpu_Warping', 'Gpu_Project2VirtualPlane' and 'Gpu_Plane_Intersection' correspond to the fellow-chart in Fig. 9.

The images of all virtual cameras initially get projected onto a temporary inertial plane. This part of the algorithm (labelled as 'Gpu_Plane_Intersection') is illustrated in Fig. 8 to demonstrate the intersection for an exemplary inertial-plane π_h among the inertial planes. $\pi_h^{(V_i)}$ indicates the temporary inertial-plane corresponding to the virtual camera V_i. Then the corresponding cells of all temporary inertial-planes are fused using an AND operator in order to provide the final registration on the inertial-plane π_h. (m and n indicate the indices of a cell). Note that the images are considered as binary.

Fig. 10 depicts another view of the algorithm. The parts colored in yellow are implemented on CUDA. Fig. 9 demonstrates the flowchart of the parallel implementation using CUDA. In the beginning the images are grabbed and then the silhouettes are extracted. After that the silhouettes are loaded on the GPU memory in order to be processed by CUDA. The loaded images on GPU memory are warped to generate the images of virtual cameras (labelled as VirImgGen). After having the images of the virtual cameras generated, the images are projected on the different inertial-planes in order to register the 3D data on them (labelled as GPU_Project2VirtualPlane). Once images of all cameras get projected onto the inertial-planes, a pixel-wise AND operator is applied to them in order to obtain the intersections (labelled as Gpu_Plane_Intersection). In this point the 3D volumetric reconstruction has been obtained. Eventually the registered data are passed to a visualizer to show the result. The processes labelled by VirImgGen, GPU_Project2VirtualPlane and Gpu_Plane_Intersection are the part which are done on CUDA using a parallel implementation.

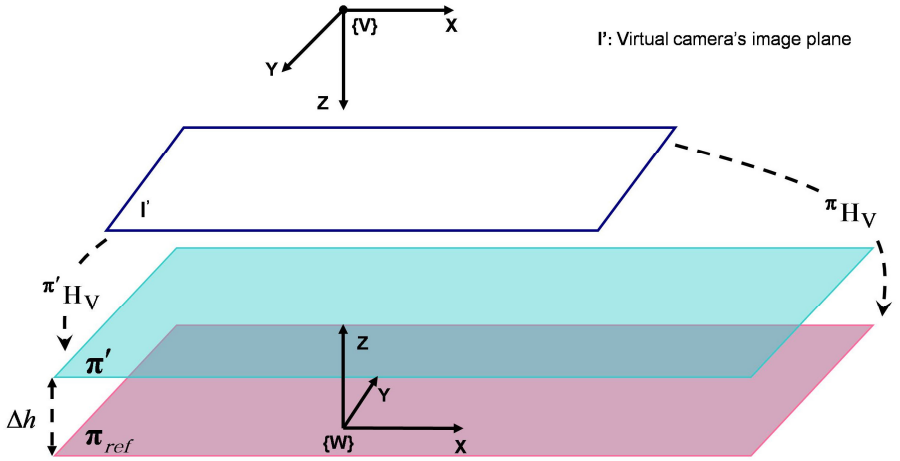

Fig. 6 Extending homography for planes parallel to π_{ref}. ${}^{\pi}H_V$ is the available homography matrix among virtual image plane I' and the first inertial-based virtual plane π_{ref}. π' is another inertial-based virtual plane, parallel to π_{ref}. Δh is the distance among π and π'. The idea is to obtain ${}^{\pi'}H_V$, the homography between the image plane and π', having the ${}^{\pi}H_V$ and Δh (see Eq. (14)).

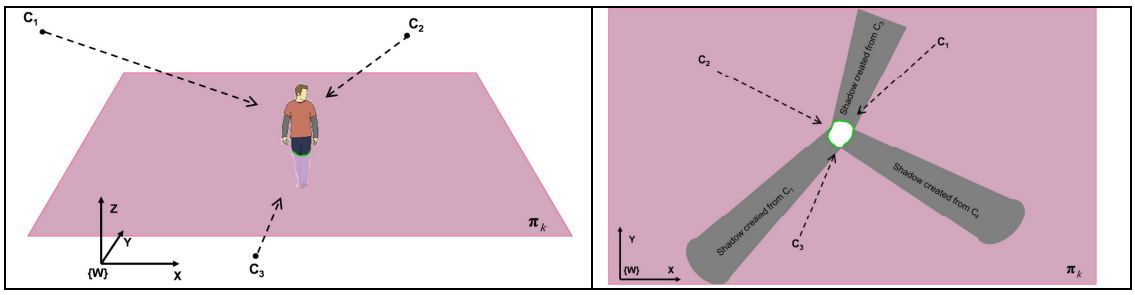

Fig. 7 Illustration of the registration using homography concept. Left: A scene including a human is depicted. π_k is one inertial-based virtual world plane. The cameras are observing the scene. Right: The registration layer (top view of the plane π_k of left figure). Each camera can be interpreted as a light source.

"Gpu_Warping"

/* generating image planes of virtual cameras */
for $i \leftarrow 1$ **to** N_c **do**
  ${}^{v_i}H_{c_i} \leftarrow K_i \, {}^{v_i}R_{c_i} \, K_i^{-1}$
  $I_{v_i} \leftarrow {}^{v_i}H_{c_i} \; I_{c_i}$
  ${}^{v_i}H_{\pi_{ref}} \leftarrow K_{c_i}[\, \hat{\mathbf{i}} \; -\hat{\mathbf{j}} \;\; \mathbf{t}_{c_i}\,]$ /* Eq. (13) */

"Gpu_Project2VirtualPlane"

/* projecting virtual images onto inertial-planes */
$h \leftarrow 0$
for $j \leftarrow 0$ **to** $N_\pi - 1$ **do**
  for $i \leftarrow 1$ **to** N_c **do**
    ${}^{\pi_h}H_{v_i} \leftarrow inv({}^{v_i}H_{\pi_{ref}} + h\, P_i\, \hat{\mathbf{k}}^T)$ /* using Eq. (14) */
    $\pi_h^{(v_i)} \leftarrow {}^{\pi_h}H_{v_i} \; I_{v_i}$
  $h \leftarrow h + \Delta h$

"Gpu_Plane_Intersection"

/* obtaining intersection of the projected virtual images for each inertial plane */
for $j \leftarrow 0$ **to** $N_\pi - 1$ **do**
  /* cell-wise binary AND. Please see Fig. 8 */
  $\pi_j \leftarrow \prod_{i=1}^{N_c} \pi_h^{(v_i)}$

return $\{\pi_0, \pi_1 \cdots \pi_{(N_\pi - 1)}\}$

Algorithm 1 Algorithm of 3D data registration using inertial-planes: First, the image plane of each virtual camera is obtained. Note that the background-subtracted images are binary. Then the image of each virtual camera is projected onto a set of inertial-planes. N_c and N_π indicate the number of cameras and number of inertial-planes, respectively. I_{c_j} and I_{v_j} respectively are the image planes of camera C_j and its corresponding virtual camera V_j. Δh is the Euclidean distance among the inertial-planes which also can be interpreted as the vertical resolution of the algorithm. (The labels 'Gpu_Warping', 'Gpu_Project2VirtualPlane' and 'Gpu_Plane_Intersection' correspond to the flowchart in Fig. 9)

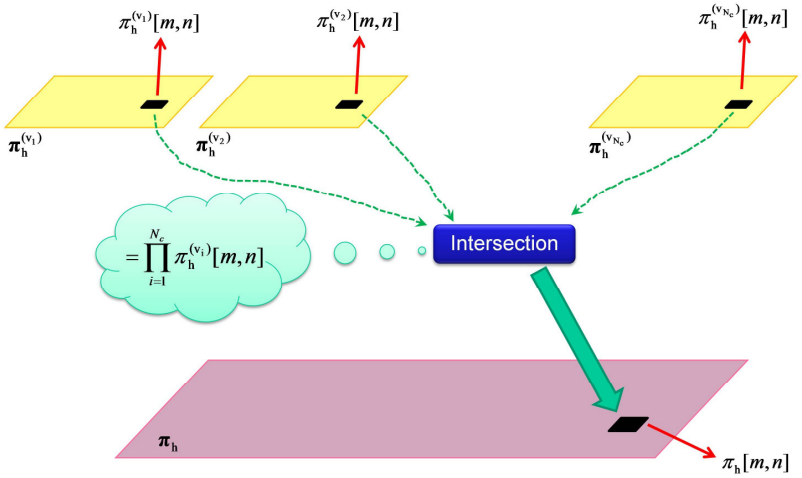

Fig. 8 Cell-wise intersection of the projections of the virtual images onto an exemplary inertial-plane π_h: Firstly the images of all virtual cameras get projected onto a temporary inertial plane. $\pi_h^{(v_j)}$ indicates the temporary inertial-plane corresponding to the virtual camera V_j. Then the corresponding cells of all temporary inertial-planes are fused using an AND operator in order to provide the final registration on the inertial-plane π_h. (m and n indicate the indices of a cell). Note that the images are considered as binary.

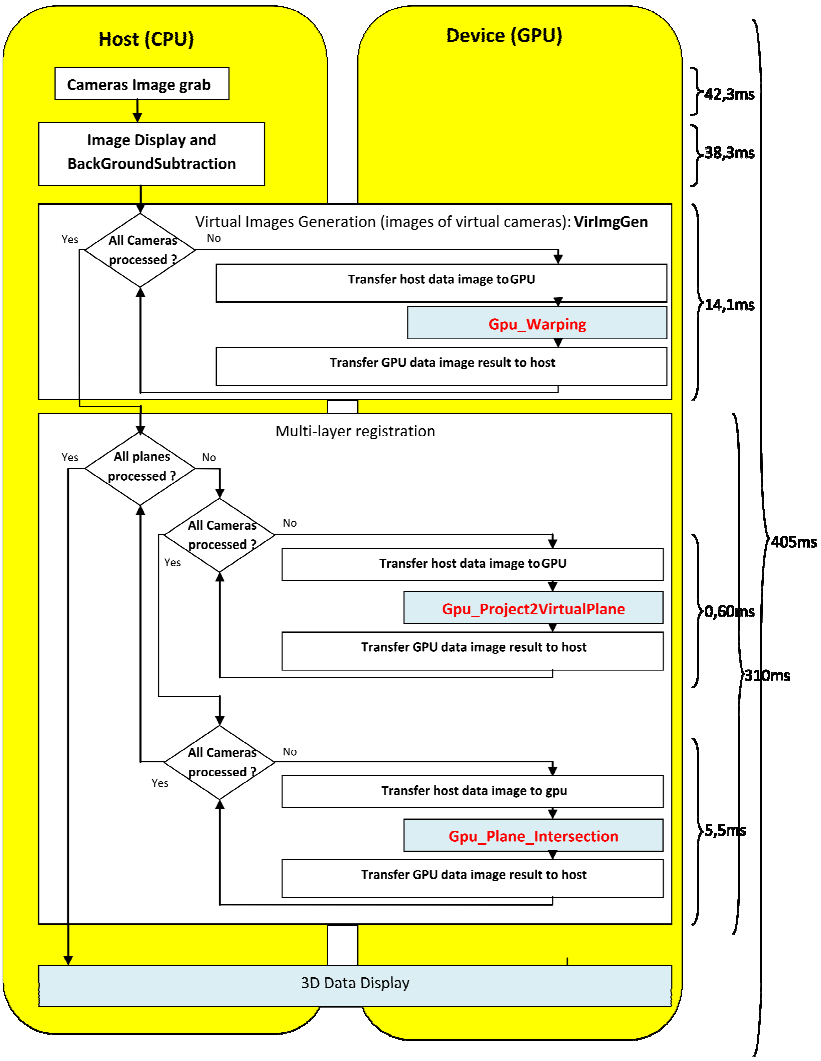

Fig. 9 Flowchart of CUDA implementation of the proposed inertial-based 3D reconstruction. In the beginning the images are grabbed and then the silhouettes are extracted. After that the silhouettes are loaded on the GPU memory. The loaded images on GPU memory are warped to generate the images of virtual cameras (VirImgGen). This part for each camera is done using parallel implementation. After having the images of the virtual cameras generated, the images are projected on the different inertial-planes in order to register the 3D data on them (GPU_Project2VirtualPlane). Once images of all cameras get projected onto the inertial-planes, a pixel-wise AND operator is applied to them in order to obtain the intersections. In this point the 3D volumetric reconstruction has been obtained. Eventually the registered data are passed to a visualizer to display the result.

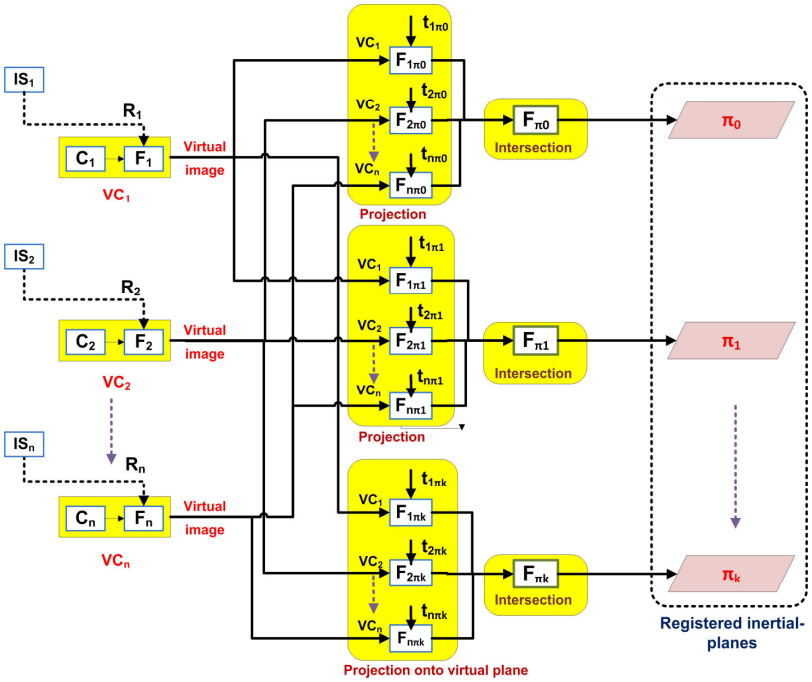

Fig. 10: The architecture corresponding to the proposed algorithm. The parts coloured in yellow are implemented on CUDA.

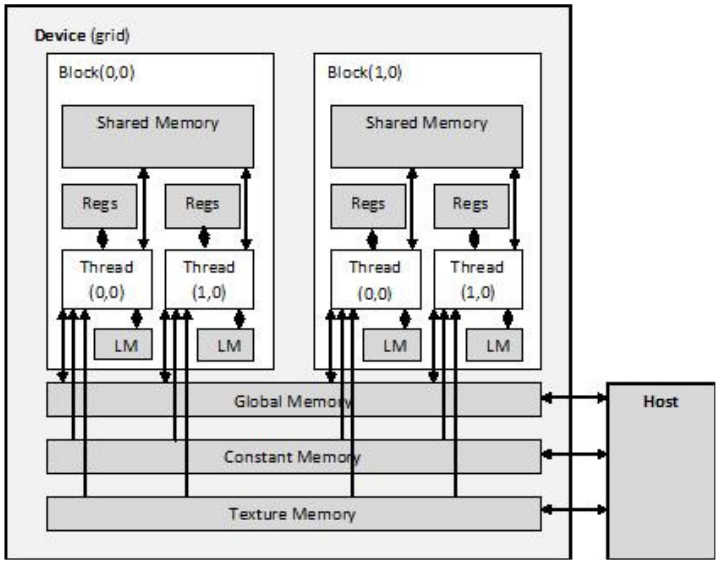

Fig. 11 CUDA architecture.

3. Experiments

3.1 Infrastructure

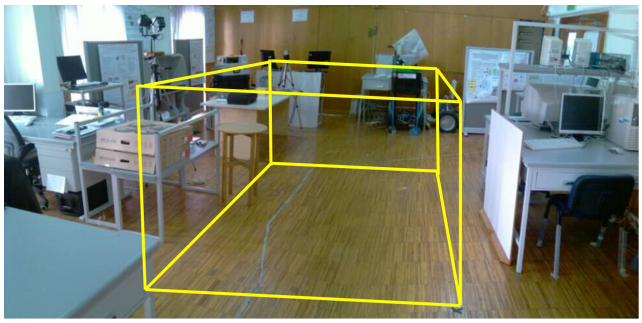

Fig. 12 The scene used in the 3D reconstruction experiments. The superimposed area indicates where all cameras have overlap in their field of view.

Fig. 12 shows the smart-room of the laboratory of mobile robotic in the University of Coimbra [40], used in our experiments. The superimposed area in this figure is observed by a camera network. The cameras are AVT Prosilica GC650C GigE Color [41], synchronized by hardware. Each camera is rigidly coupled with an IS (we used Xsens MTx [42]). The purpose of using IS is to have 3D orientation with respect to earth, obtain virtual camera and define virtual horizontal planes. First the intrinsic parameters of the cameras are estimated using Bouguet Camera Calibration Toolbox [43] and then Camera Inertial Calibration Toolbox [44] is used for the sake of extrinsic calibration between the camera and IS (to estimate ${}^{C}R_{IS}$). For extrinsic calibration of cameras, a method proposed in our previous work [30] is used. After acquiring image from each camera, a color-based background subtraction step is performed. To east the background subtraction, the person is dressed in red and his silhouette is separated from the background through color segmentation using the HSV (hue, saturation, value) model. This model is less sensible to illumination changing conditions [45,46]. A 1-D Hue histogram is sampled from the human area and stored for future use. During frame acquisition, the stored color histogram is used as a model, or look-up table, to convert incoming video pixels to a corresponding probability of body image. Using this method, probabilities range in discrete steps from zero (0.0) to the maximum probability pixel (1.0). Later it is multiplied by a binary mask.

The reconstruction algorithm was developed using the C++ language, OpenCV library [42] and NVIDIA's CUDA software [43] for Ubuntu Linux v10.10. The processing unit responsible for the entire sensory and vision algorithm (including CUDA processing) is composed by a PC (Intel Core2 Quad processor Q9400, 6 MB Cache, 4 GB RAM, 1333 MHz and a PCI-Express NVIDIA GeForce 9800 GTX+).

3.2 Reconstruction results

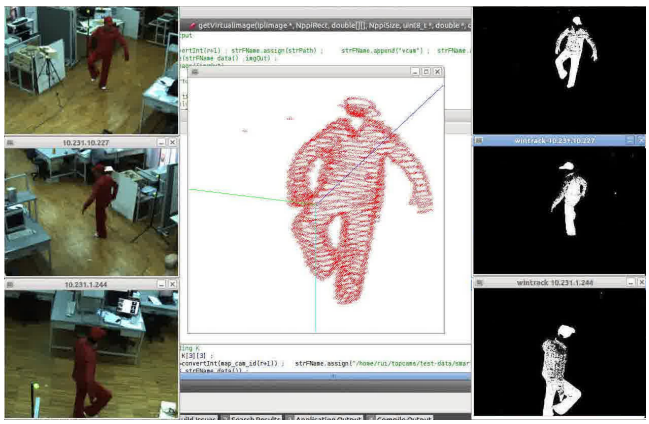

Fig. 13 Results of 3D volumetric reconstruction using the proposed framework: The camera images before and after background subtraction (silhouette) are respectively shown in the left and right columns. The result of volumetric reconstruction using the silhouette is illustrated in the middle. A network of IS-camera is used to observe the scene. 48 inertial-planes are used to register 3D data from the scene. The interval distance among two consecutive inertial-plane is $5cm$.

A set of experiments have been carried out using the proposed inertial-based 3D reconstruction method by a GPU-based implementation. 24 samples are demonstrated

in Fig. 14 and 15 where an acting person is reconstructed in 3D. One of the samples is seperatly shown in Fig. 13 in order to have a more detailed view. In these examples, 48 inertial-planes are used for the purpose of 3D data registration. The interval distance among two consecutive inertial-plane is $5cm$.

Fig. 14 Results of 3D volumetric reconstruction using the proposed framework: 12 samples have been illustrated. In each sample, the camera images before and after background subtraction (silhouette) are respectively shown in the left and right columns. The result of volumetric reconstruction using the silhouette is illustrated in the middle column for each sample. A network of IS-camera is used to observe the scene. 48 inertial-planes are used to register 3D data from the scene. The interval distance among two consecutive inertial-plane is $50mm$.

Although the area of the scene in these experiments is small however in the computation the area is considered as $384 \times 384\ cm^2$ which is relatively large. Using a parallel implementation of the algorithm (using GPU), we managed to have a frequency reconstruction close to $2.5Hz$ for the mentioned area (using the hardware stated in sub-section 3.1). The number of layers and their intervals can be adjusted depending to the application and available hardware.

3.2.1 Statistical analysis on the processing times

Fig. 15 Results of 3D volumetric reconstruction using the proposed framework: 12 samples have been illustrated. In each sample, the camera images before and after background subtraction (silhouette) are respectively shown in the left and right columns. The result of volumetric reconstruction using the silhouette is illustrated in the middle column for each sample. A network of IS-camera is used to observe the scene. 48 inertial-planes are used to register 3D data from the scene. The interval distance among two consecutive inertial-plane is $50mm$.

Some statistics are carried out in order to show the time

which each part of the algorithm takes to run. In Fig. 9, processing time for each part of the algorithm is imprinted. The times refer to the case where 48 inertial-planes, each one having a size of $384 \times 384\ cm^2$, have been used. The infrastructure and hardware are as stated in sub-section 3.1. The total processing time for a complete 3D reconstruction is $405\ ms$ which leads to have a frequency close to $2.5Hz$.

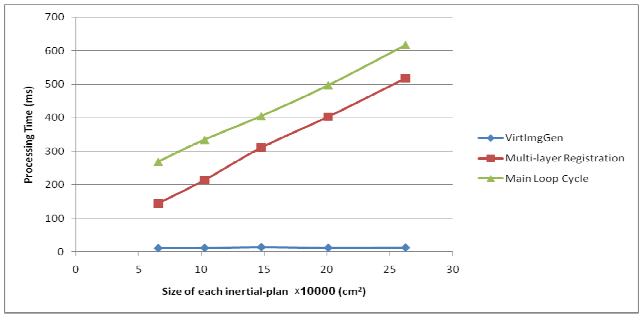

Fig. 16 Average processing times in ms for different size of inertial-planes. The notations are related to the flowchart shown in Fig. 9. Number of 2D inertial-planes used in this statistic is 48.

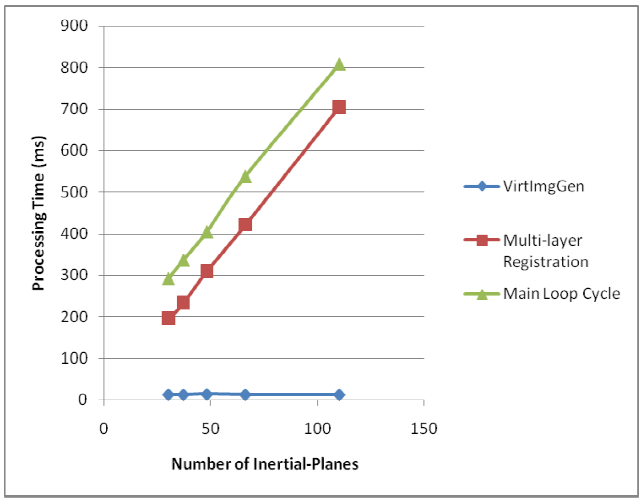

Fig. 17 Average processing times in ms for different number of inertial-planes. The notations are related to the flowchart shown in Fig. 9. The size of each 2D inertial-planes used in this statistic is $384 \times 384\ cm^2$.

Fig. 16 depicts the average processing time in ms for different size of inertial-planes (the scale is 10^4 cm^2). The number of inertial-planes in this experiments is a constant equal to 48. The blue line demonstrates the processing time for generating the images of virtual cameras. Since the number of cameras is fixed in all tests, the execution time for that is almost constant. The red line indicates a part where images of all virtual cameras get projected onto a set of inertial-planes. Eventually the total processing time is shown in green color. As it is visible in the diagram, the processing time has a linear proportion related to size (area) of inertial-planes.

Another diagram showing the processing time versus time of inertial-planes is shown in Fig. 17. The size of inertial-planes (they are equal in the sizes) is considered as a constant equal to $384 \times 384\ cm^2$. Similar to Fig. 16, the colors blue, red and green respectively indicate the processing time of virtual images generation, projection of generated virtual imaged onto a set of inertial-planes and the total algorithm cycle. Also in this diagram the processing time has a linear proportion related to number of allocated inertial-planes.

3.3 Extension for mobile sensor

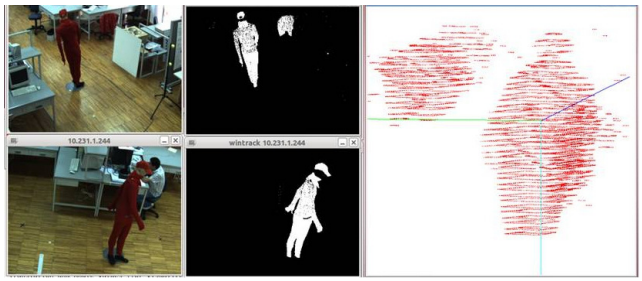

Fig. 18 Mobile sensor experiment: Result of 3D reconstruction when just two IS-camera couples are used. The other cameras are intentionally blinded. The result is shown in the right column. Because of lack of views, the details are not clear and moreover a ghost object has appeared.

Fig. 19 Result of 3D reconstruction when a mobile sensor is augmented to the network (corresponding to Fig. 18); In order to have more details of the scene, a mobile sensor is navigated close to the manikin and its view is integrated as a new node in the network. The left two columns are the images corresponding to the two fixed cameras and the third column from left is the image corresponding to the mobile camera. The results of the 3D reconstruction by using two fixed IS-camera couples and a new augmented couple are demonstrated in the right column.

The previously shown experiments were carried out by using static sensors. In some scenarios, it would be very useful to have a mobile sensor which could move inside the scene and collect data from an arbitrary point of view. The data provided by it can be used as a regular node of the sensor network. Such a mobile sensor has two main advantages: Firstly, always it is not possible to have many cameras (specially in large areas) to have all details of the different parts of the scene. Secondly, in some cases one of the main nodes (IS-camera couples) could be occluded or in any reason stop to work. In such situations, a mobile sensor could approach to an appropriate position in the scene, gather and transmit close-view information to the infrastructure. The proposed framework has the ability to integrate the data coming from a mobile sensor. The localization and navigation of a mobile sensor are the two old topics in the area of robotics and computer vision and there can be found many papers in the literatures which proposed different solutions for these problems. Therefore we do not enter in these areas and just assume that we have these techniques already available. In following, an experiment is provided to show the advantage of using a mobile sensor. In order to localize the mobile sensor, the method proposed in [30] is used. Fig. 18 shows a case where just two cameras from the infrastructure is used for the 3D reconstruction of a manikin (we intentionally blinded the other cameras). As can be seen, in such situations that there is not enough views to see the scene, the result of 3D

reconstruction is not good enough. As seen, there is no enough detail about the reconstructed person and moreover a ghost object has appeared as noise. In order to have more details of the scene, a mobile sensor is navigated close to the manikin. Then after localizing the mobile sensor, its view is integrated as a new node in the network.

The results of the 3D reconstruction by using two fixed IS-camera couples and a new added couple are demonstrated in fig. 19. This figure shows the advantage of having a mobile sensor which could cooperate with the infrastructure.

3.4 Discussion

A set of experiments to demonstrate the applicability and effectiveness of the proposed reconstruction method has been provided. The provided analytical diagrams show acceptable processing times for performing the fully reconstruction of human body within a scene using different parameters. In our experiments although the ground is planar however this ground is not used for estimation of homography matrices. Instead of using planar ground assumption we used the inertial planes to define a virtual ground plane. Many of the introduced papers in the state-of-the-art assumed to have a planar ground such a way that it has to be possible to mark some points on the ground in order to estimate the homography matrix among the image plane and ground [2,5,10,11,49]. This is not always possible for two main reasons. Firstly in some outdoor scenarios it is not possible to have a flat ground plane. Secondly in some textured grounds it is not possible to use (or mark) a set of points from the ground with known geometry relation among them. Some other authors assumed to have a set of vertical parallel lines in the scene and their images in order to estimate the vertical vanishing point [7,50]. This assumption can not be satisfied as well in some scenarios because of not availability of enough vertical lines in the scene, specially for not man-made scenes.

4. Conclusion

Human volumetric reconstruction is demanding by many applications such as human-behaviour understanding, smart-room, health-care, surveillance etc. Nowadays, camera network is frequently deployed for public or even private observations for different purposes depending to the application. Recently, IS is becoming much cheaper and more available. Even many smart phones can be found equipped in both IS and camera. Taking advantage of this, we used a network of IS-camera couples to observe the scene and then a method for 3D reconstruction of a person using inertial data and with no planar ground assumption was proposed. In order to achieve a real-time execution, a parallel processing architecture was proposed and implemented on CUDA. The 3D reconstructions of a person acting in the scene in different gestures are quite promising. The experiments demonstrated the applicability and effectiveness of the proposed approach for many applications.

Acknowledgments:

Hadi Ali Akbarpour is supported by the FCT (Portuguese Fundation for Science and Technology). The authors would like to thank Joao Quintas and Amilcar Ferreira for their helps in the preparation of the smart-room and data acquisition.

References

1. H. Aliakbarpour J. Dias (2010) IMU-aided 3D Reconstruction based on Multiple Virtual Planes, *DICTA'10 (the Australian Pattern Recognition and Computer Vision Society Conference), IEEE Pr*, 1-3 December, Sydney, Australia.
2. S. M. Khan, P. Yan, M. Shah (2007) A Homographic Framework for the Fusion of Multi-view Silhouettes, *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference*.
3. B. Michoud, E. Guillou, S. Bouakaz (2007) Real-Time and Markerless 3D Human Motion Capture Using Multiple Views, *Human Motion-Understanding, Modeling, Capture and Animation, Springer Berlin/Heidelberg*, **4814/2007**:88-103
4. Q. Zhang, H. Wang, S. Wei (2003) A NEW ALGORITHM FOR 3D PROJECTIVE RECONSTRUCTION BASED ON INFINITE HOMOGRAPHY. *Machine Learning and Cybernetics, 2003 International Conference on, IEEE.*
5. Z. Zhang, A. R. Hanson (1996) 3D Reconstruction Based on Homography Mapping, *In ARPA Image Understanding Workshop.*
6. M. Sormann, C. Zach, J. Bauer, K. Karner, H. Bishof (2007) Watertight Multi-view Reconstruction Based on Volumetric Graph-Cuts. *In Ersball, Bjarne and Pedersen, Kim, editors, Image Analysis in Lecture Notes in Computer Science, Springer Berlin, Heidelberg,* 393-402.
7. P. Lai, A. Yilmaz (2008) PROJECTIVE RECONSTRUCTION OF BUILDING SHAPE FROM SILHOUETTE IMAGES ACQUIRED FROM UNCALIBRATED CAMERAS. *ISPRS Congress Beijing 2008, Proceedings of Commission III.*
8. T. Feldmann, I. Mihailidis, S. Schulz, D. Paulus, A. Worner (2010) Online Full Body Human Motion Tracking Based on Dense Volumetric 3DÂ Reconstructions from Multi Camera Setups. In Dillmann, Rudiger and Beyerer, Jurgen and Hanebeck, Uwe and Schultz, Tanja, editors, *KI 2010, Advances in Artificial Intelligence in Lecture Notes in Computer Science, Springer Berlin/Heidelberg,* 74-81.
9. R. Guerchouche, O. Bernier, T. Zaharia (2008) Multiresolution volumetric 3D object reconstruction for collaborative interactions. *Pattern Recognition and Image Analysis*, **18**:621-637. 10.1134/S1054661808040147.
10. T. Azevedo, J. Tavares, M. Vaz (2009) 3D Object Reconstruction from Uncalibrated Images Using an Off-the-Shelf Camera. *Advances in Computational Vision and Medical Image Processing; in series of Computational Methods in Applied Sciences, Springer Netherlands*, **13**:117-136, Universidade do Porto.
11. H. Lee, A. Yilmaz (2010) 3D RECONSTRUCTION USING PHOTO CONSISTENCY FROM UNCALIBRATED MULTIPLE VIEWS. *VISAPP 2010 - The International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*,
12. B. Zhang, Y.F. Li (2005) An efficient method for dynamic calibration and 3D reconstruction using homographic transformation. *Sensors and Actuators A: Physical*,

119(2):349 - 357
13. H. Lin, J. Wu (2008) 3D Reconstruction by Combining Shape from Silhouette with Stereo. *IEEE*.
14. B. Michoud, S. Bouakaz, E. Guillou. H. Briceno (2008) Largest Silhouette-Equivalent Volume for 3D Shapes Modeling without Ghost Object. *M2SFA2 2008: Workshop on Multi-camera and Multi-modal Sensor Fusion*, Marseille, France.
15. H. Aliakbarpour, J. Dias. (2010) Human Silhouette Volume Reconstruction Using a Gravity-based Virtual Camera Network. *Proceedings of the 13th International Conference on Information Fusion*, 26-29 July 2010 EICC Edinburgh, UK.
16. A. Calbi, C. Regazzoni, L. Marcenaro (2006) Dynamic Scene Reconstruction for Efficient Remote Surveillance. *IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS'06)*.
17. J. Franco, E. Boyer (2005) Fusion of Multi-View Silhouette Cues Using a Space Occupancy Grid. *Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV05)*.
18. J. F. Ferreira, J. Lobo, J. Dias (2010) Bayesian real-time perception algorithms on GPU --- Real-time implementation of Bayesian models for multimodal perception using CUDA. *Journal of Real-Time Image Processing*, Special Issue.
19. L. Almeida, P. Menezes, J. Dias (2011) Stereo Vision Head Vergence Using GPU Cepstral Filtering. *Proceedings of the Fifth International Conference on Computer Vision Theory and Applications (VISAPP)*, Vilamoura, Algarve, Portugal, March, 5-7.
20. A. Griesser, S. D. Roeck, A. Neubeck, L. V. Gool (2005) Gpu-based foreground-background segmentation using an extended colinearity criterion. *In Proc. Vision, Modeling, and Visualization (VMV) 2005*. Amsterdam, The Netherlands: IOS, Nov. 2005., 319-326
21. G. Ziegler (2010) GPU Data Structures for Graphics and Vision. *PhD thesis*, Max-Planck-Institut fÃ¼r Informatik.
22. C. Nitschke, A. Nakazawa, H. Takemura (2007) Real-Time Space Carving Using Graphics Hardware. *IEICE - Trans. Inf. Syst.*, E90-D:1175--1184
23. S.S. Stone, J.P. Haldar, S.C. Tsao, W.-m.W. Hwu, B.P. Sutton, Z.-P. Liang (2008) Accelerating advanced MRI reconstructions on GPUs. *Journal of Parallel and Distributed Computing*, **68**(10):1307-1318, General-Purpose Processing using Graphics Processing Units.
24. W. Waizenegger, I. Feldmann, P. Eisert, P. Kauff (2009) Parallel high resolution real-time Visual Hull on GPU. *Image Processing (ICIP), 2009 16th IEEE International Conference on*, 430-4304
25. S. Yous, H. Laga, M. Kidode, K. Chihara (2007) GPU-based shape from silhouettes. *Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia in GRAPHITE '07*, 71-77, New York, NY, USA, 2007. ACM.
26. D. Knoblauch, F. Kuester (2009) Focused Volumetric Visual Hull with Color Extraction. In Bebis, George and Boyle, Richard and Parvin, Bahram and Koracin, Darko and Kuno, Yoshinori and Wang, Junxian and Pajarola, Renato and Lindstrom, Peter and Hinkenjann, Andre and Encarnacao, Miguel and Silva, Claudio and Coming, Daniel, editors, *Advances in Visual Computing in Lecture Notes in Computer Science*, *Springer Berlin-Heidelberg,*208-217
27. A. Ladikos, S. Benhimane, N. Navab (2008) Efficient visual hull computation for real-time 3D reconstruction using CUDA. *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08. IEEE Computer Society Conference on*, 1-8.
28. M. Yguel, O. Aycard, C. Laugier (2006) Efficient GPU-based Construction of Occupancy Grids Using several Laser Range-finders. Oct. 2006. *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*.
29. M. Kalantari, A. Hashemi, F. Jung, J. Guedon (2011) A New Solution to the Relative Orientation Problem Using Only 3 Points and the Vertical Direction. *Journal of Mathematical Imaging and Vision*, **39**:259-268
30. H. Aliakbarpour, J. Dias (2011) Inertial-Visual Fusion For Camera Network Calibration. *IEEE 9th International Conference on Industrial Informatics (INDIN 2011).*
31. J. Lobo, J. Dias (2003) Vision and inertial sensor cooperation using gravity as a vertical reference. **Pattern Analysis and Machine Intelligence, IEEE Transactions on**, **25**(12):1597-1608
32. M. Labrie, P. Hebert, (2007) Efficient camera motion and 3D recovery using an inertial sensor. *Computer and Robot Vision, 2007. CRV '07. Fourth Canadian Conference on*, 55 - 62
33. T. Okatani, K. Deguchi (2002) Robust estimation of camera translation between two images using a camera with a 3D orientation sensor. *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, **1**:275 - 278
34. M. A. Brodie, A. Walmsley, W. Page (2008) The static accuracy and calibration of inertial measurement units for 3D orientation. *Computer Methods in Biomechanics and Biomedical Engineering*, **11**:641-648
35. R. Hartley, A. Zisserman (2003) Multiple View Geometry in Computer Vision. *CAMBRIDGE UNIVERSITY PRESS*,
36. Y. Ma, S. Soatta, J. Kosecka, S. S. Sastry (2004) *An invitation to 3D vision. Springer*.
37. L. G. B. Mirisola, J. Dias, A. Traca de Almeida (2007) Trajectory Recovery and 3D Mapping from Rotation-Compensated Imagery for an Airship. *Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems San Diego*, CA, USA, Oct 29 - Nov 2, 2007.
38. L. G. B. Mirisola (2009) Exploiting attitude sensing in vision-based navigation, mapping and tracking including results from an airship. *PhD thesis*.
39. L. G. B. Mirisola, J. M. M. Dias (2007) Exploiting inertial sensing in mosaicing and visual navigation. *In 6th IFAC Symposium on Inteligent Autonomous Vehicles (IAV07)*, Toulouse, France, Sep. 2007.
40. MRL, *http://paloma.isr.uc.pt/mrl/*.
41. Prosilica, *http://www.1stvision.com/cameras/Prosilica/GC650-GC650C.html.*
42. Xsens Motion Technologies. http://www.xsens.com.
43. J. Bouguet (2003) Camera Calibration Toolbox for Matlab. *www.vision.caltech.edu/bouguetj.*
44. J. Lobo, J. Dias (2007) Relative Pose Calibration Between Visual and Inertial Sensors. *International Journal of Robotics Research, Special Issue 2nd Workshop on Integration of Vision and Inertial Sensors*, **26**:561-575
45. P. Kakumanu, S. Makrogiannis, N. Bourbakis (2007) A survey of skin-color modeling and detection methods. *Pattern Recogn*, **40**:1106-1122
46. G. R. Bradski (1998) Computer Vision Face Tracking For Use in a Perceptual User Interface. *Intel Technology Journal*, *(Q2)*.
47. OpenCV. *http://opencv.willowgarage.com/*.
48. NVIDIA. *http://www.nvidia.com/*.
49. T. Wada, X. Wu, S. Tokaim. T. Matsuyama (2000) Homography Based Parallel Volume Intersection: Toward Real-Time Volume Reconstruction Using Active Cameras. *Computer Architectures for Machine Perception, 2000. Proceedings. Fifth IEEE International Workshop* on 11-13 Sept. 2000, 331-339
50. P. Lai, A. Yilmaz (2008) Efficient object shape recovery via slicing planes. *Computer Vision and Pattern Recognition,*

2008. CVPR 2008. IEEE Conference on, 1-6