Featureless omnidirectional vision-based control of non-holonomic mobile robot

Youssef Alj, Guillaume Caron

The 12th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI 2015)
October 28 ∼ 30, 2015 / KINTEX, Goyang city, Korea

Featureless omnidirectional vision-based control of non-holonomic mobile robot

Youssef Alj, Guillaume Caron

Université de Picardie Jules Verne, "Modélisation, Information & Systèmes Laboratory", 33 rue Saint-Leu - 80039 Amiens Cedex 1 - France
(E-mail :youssef_alj@hotmail.com ; guillaume.caron@u-picardie.fr)

Abstract - This paper proposes featureless algorithms to address complex maneuvers for non-holonomic mobile robots. Using a single omnidirectional camera, the robot reaches the target position using an omnidirectional photometric visual servoing algorithm that simultaneously controls translation and rotation. Evaluation of the positioning task was performed on eight different positions on a circle and the error between the final position and the target one is measured.

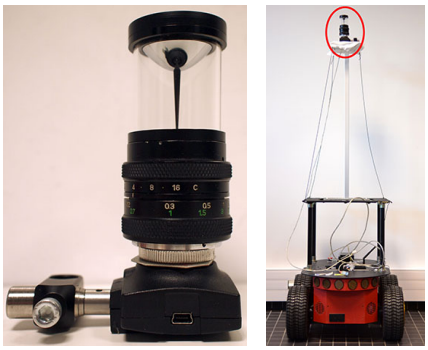
Keywords - visual servoing, omnidirectional camera, featureless, non-holonomic mobile robots.

1. Introduction

The goal of this work is to extend the omnidirectional photometric visual servoing [4] for complex maneuvers in mobile robot positioning and robust visual path following.

Controlling a robot with vision as input finds its interest in the acquisition of a lot of information in one image. This passive sensor is not range-limited, or only by the environment. While the field of view of a standard perspective camera is limited, an omnidirectional camera increases the field of view to 180 degrees, only changing the camera optics. Two families of such optics exist : the dioptric ones (fisheye lens) and the catadioptric ones (combination of lens and quadratic mirror). They allow acquiring a panoramic image in one shot, when their main optical axis is oriented vertically, a common use of these vision sensors on mobile robots (Fig. 1).

Vision-based robot control can be classified in two classes : feature-based or featureless. Contrary to visual feature-based control, featureless control has the interest of avoiding geometric features detection and matching, making the controller closer to data acquired by the sensor, that are pixel intensities of the image acquired by the camera : the photometry. Featureless or not, the vision-based control of a robot is described under the visual servoing framework [5]. It models the relationship between the motion of the camera and visual features in the image, leading to a kinematic proportional controller in basic formulations. Many visual features have been considered for visual servoing, as a point [8], a line [1] or sets of points [2], to only mention some of the ones introduced for omnidirectional vision in mobile robotics. Featureless, or photometric, visual servoing in omnidirectional vision for mobile robotics [4] considers the set of intensities of a desired image, *i.e.* acquired at the desired location of the mobile robot, as input of the mobile robot

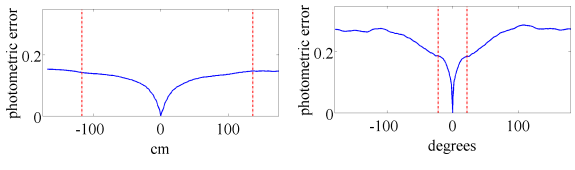


(a) (b)

Fig. 1 (a) The considered catadioptric camera, with a IDS uEye box ; (b) The camera vertically mounted and centered on a Pioneer 3-AT mobile robot.

control law [6]. Thus, visual servoing designs the control law to drive the robot minimizing the photometric error, *i.e.* the sum of squared differences (SSD) or other metrics (e.g. Hausdorff distance [9]), between the current image intensities and the desired ones.

However, every interest of featureless control has still not been shown because existing photometric visual servoing, even involving omnidirectional vision, has limitations, even for the most studied mobile robot kind, that is the unicycle robot. Indeed, such so called non-holonomic platform can not do instantaneously lateral motion leading to the need of maneuvers, if a lateral motion would be needed to achieve the mobile robot mission. Existing omnidirectional photometric visual servoing works well if such a motion is not needed but as soon as the mobile robot needs to rotate twenty degrees to reach its desired pose, the system may not converge on it. The reason will be clearly described in Section B. , but Figure 2(b) highlights it coarsely : the cost function, that is the photometric error, is much more sensitive to the rotational degree of freedom of the unicycle robot rather than its translational one (convergence area of 2.50 m width in



(a) (b)

Fig. 2 On an example : (a) Cost function for single axis pure translation ; (b) Cost function for pure rotation

Fig. 2(a)).

The contribution of the paper is to provide a solution to the latter issue, still directly using image intensities. In a few words, the core idea of the proposed method is to consider the re-orientation of the desired image with respect to the current one, thanks to visual compass, to overcome the sensitiveness of photometric visual servoing to rotation around the camera optical axis. Doing so has impacts on the control law that is to be described in Section 3. for the positioning issue, after reminding existing tools in Section 2A final experiment and a conclusion end the paper.

2. Existing tools

2.1 Notations

In the next sections we will use the following notations related to omnidirectional images :

- The desired image : $\mathbf{I}^*$; $\mathbf{I}^*_{\mathcal{S}}$ its spherical representation.
- The desired image rotated by θ : $\mathbf{I}^*_\theta$.
- The current image $\mathbf{I}$; $\mathbf{I}_{\mathcal{S}}$ its spherical representation.

2.2 Omnidirectional photometric visual servoing : the unicycle robot case

A. Method

Omnidirectional photometric visual servoing for a unicycle mobile robot defines its kinematic control inputs as :

$$\dot{\mathbf{q}} = (\upsilon_X, \omega_Z), \tag{1}$$

minimizing the photometric error (SSD) between the current and the desired omnidirectional image. Considering the camera and the robot axes are the same, that is the camera is at the robot center of rotation, its X-axis is pointing forward the robot and its Z-axis pointing up, the transformation matrix between the robot velocity $\dot{\mathbf{q}}$ and the camera velocity $\mathbf{v}$, can be neglected and the Levenberg-Marquardt-like control law is defined as [4] :

$$\dot{\mathbf{q}} = -\lambda \left(\mathbf{H} + \mu \operatorname{diag}(\mathbf{H})\right)^{-1} \mathbf{L}_{\mathbf{I}_{\mathcal{S}}}^{\top} \left(\mathbf{I}_{\mathcal{S}}(\mathbf{r}) - \mathbf{I}^*_{\mathcal{S}}(\mathbf{r}^*)\right) \tag{2}$$

with λ a gain, μ, the damping parameter,
$\mathbf{r} = [t_X, t_Y, t_Z, \phi * w_X, \phi * w_Y, \phi * w_Z]$, with $\mathbf{w} = [w_X, w_Y, w_Z]$ the 3D axis of rotation of unit norm and ϕ the angle, that are the six degrees of freedom (dof) camera pose (position and orientation) parameters. Furthermore, in Equation (2), $\mathbf{H} = \mathbf{L}_{\mathbf{I}_{\mathcal{S}}}^{\top} \mathbf{L}_{\mathbf{I}_{\mathcal{S}}}$, is the omnidirectional photometric Hessian, considering :

$$\mathbf{L}_{\mathbf{I}_{\mathcal{S}}} = \begin{bmatrix} \vdots \\ \mathbf{L}_{\mathbf{I}_{\mathcal{S}}} \\ \vdots \end{bmatrix}, \text{ with } \mathbf{L}_{\mathbf{I}_{\mathcal{S}}} = -\nabla \mathbf{I}_{\mathcal{S}}^{\top} \mathbf{L}_{\mathbf{X}_{\mathcal{S}}}, \tag{3}$$

is the interaction matrix related to luminance of the spherical image $\mathbf{I}_{\mathcal{S}}$, representing the omnidirectional image, at cartesian spherical coordinates $\mathbf{X}_{\mathcal{S}} = (X_{\mathcal{S}}, Y_{\mathcal{S}}, Z_{\mathcal{S}})$ corresponding to a pixel. The latter coordinates and spherical representation of the image are obtained considering the invert of the unified projection model [3] :

$$\mathbf{X}_{\mathcal{S}} = \begin{pmatrix} \frac{\xi + \sqrt{1+(1-\xi^2)(x^2+y^2)}}{x^2+y^2+1} x \\ \frac{\xi + \sqrt{1+(1-\xi^2)(x^2+y^2)}}{x^2+y^2+1} y \\ \frac{\xi + \sqrt{1+(1-\xi^2)(x^2+y^2)}}{x^2+y^2+1} - \xi \end{pmatrix}, \tag{4}$$

where (x, y) are the coordinates of a normalized image point, obtained from pixel coordinates (u, v) and inverse intrinsic parameters, including the above mentioned ξ, a specific parameter of the unified projection model.

In Equation (3), $\nabla \mathbf{I}_{\mathcal{S}}$ are the cartesian spherical gradients introduced in [4] and the interaction matrix $\mathbf{L}_{\mathbf{X}_{\mathcal{S}}}$ is similar to the Jacobian of $\mathbf{X}_{\mathcal{S}}$ with respect to the camera pose parameters $\mathbf{r}$. In the general case, $\mathbf{L}_{\mathbf{X}_{\mathcal{S}}}$ has six dof, so it involves 6 columns, but in the considered mobile robot case, with υ_X and ω_Z dof, we have :

$$\mathbf{L}_{\mathbf{X}_{\mathcal{S}}} = \begin{pmatrix} \frac{X_{\mathcal{S}}^2 - 1}{\rho} & Y_{\mathcal{S}} \\ \frac{X_{\mathcal{S}} Y_{\mathcal{S}}}{\rho} & -X_{\mathcal{S}} \\ \frac{X_{\mathcal{S}} Z_{\mathcal{S}}}{\rho} & 0 \end{pmatrix}. \tag{5}$$

The latter consideration of the unicycle robot two dof leads to some limitations of its maneuvering space that are described in Section B.

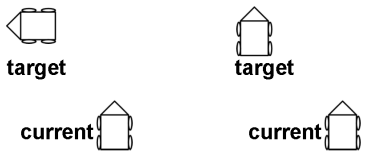
B. Limitations

Whereas the omnidirectional photometric visual servoing, recalled in Section A. , succeeds in situations where the robot motion only needs to be straightforward, or with a small change of orientation, other situations in which maneuvers are necessary may not meet success. Two common situations are identified as follows :

- First, if the needed robot motion, to reach the desired pose, would be too curvy (Fig. 3(a)), the limited convergence area highlighted in Figure 2(b) may lead to stop the robot rotation in a local minimum.
- Second, if the robot has to change its orientation during the motion for any reason, it will finally reach a pose in which its orientation is the same as the desired one but at a wrong position (Fig. 3(b)). Indeed, it woud be necessary to rotate but, by doing so, the orientation control would immediately counter that orientation change to minimize the photometric error.

Both situations involve limitations due to the control of orientation directly through the rotational degree of freedom as recalled at the end of Section A. , not the translation one that has a much larger convergence area (Fig. 2(a)). Thus, we propose to overcome these limitations by revisiting the orientation control to ensure the robot reaches the desired position and orientation, in the situations for which the existing omnidirectional photometric visual servoing succeeds and the two above mentioned situations (section 3.).

The proposed method involves a photometric visual compass that is described below.



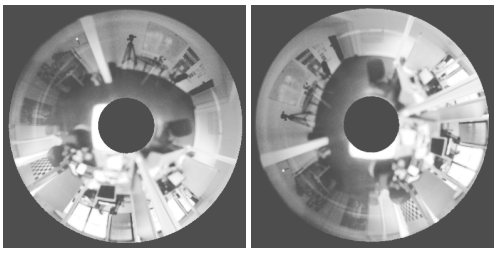
(a) (b)

Fig. 3 Two situations to address : (a) a very curvy trajectory is needed to reach the desired pose ; (b) a complex maneuver is needed to reach the desired pose

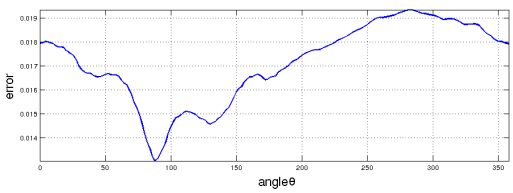
2.3 Visual compass

In the context of omnidirectional imaging, the visual compass refers to the algorithm that estimates the angle between an image $\mathbf{I}^*$ and a request image $\mathbf{I}$. In [7], the authors describe an algorithm that estimates such orientation angle. First, the two images are unwrapped into panoramic images. Assuming that the camera is perfectly vertical, a pure rotation around the camera axis will introduce a column-wise shift. The rotation angle is then computed by finding the best match between the reference image and the column-wise shift of the rotated image.

Instead of unwrapping the omnidirectional images, we propose to find the best match by directly handling the omnidirectional images. Assuming the camera's intrinsic parameters are known, $\mathbf{I}^*$ is rotated around the camera's principal point (u_0, v_0) by an increment θ. The SSD is then evaluated between the reference image, rotated by the increment $\mathbf{I}^*_\theta$, and the request image $\mathbf{I}$. The final angle is computed as the angle θ that achieves the smallest SSD between $\mathbf{I}^*_\theta$ and $\mathbf{I}$. Fig. 4 shows an example of computing such orientation angle.



(a)reference image (b)rotated image



(c)photometric error between the request image and the rotated reference image, wrt angle θ.

Fig. 4 Computing the angle between a reference and a request image using visual compass algorithm

3. OPVS for positioning

3.1 Problem overview

In order to solve the positioning task issues depicted in Figure 3, we introduce the following notations :

- $\overrightarrow{X}$ the robot direction at the current pose.
- $\overrightarrow{X^*}$ the robot direction at the target pose.
- $\overrightarrow{X^*_\theta}$ the target direction obtained by rotating $\overrightarrow{X^*}$ by θ.
- $\overrightarrow{X_p}$ the direction joining the initial to the target positions.

We denote the coincidence between two directions $\overrightarrow{X}$ and $\overrightarrow{X'}$ with the symbol $\equiv$. We can write the following : $\overrightarrow{X} \equiv \overrightarrow{X'}$ if $\overrightarrow{X}$ and $\overrightarrow{X'}$ are coincident, and $\overrightarrow{X} \not\equiv \overrightarrow{X'}$ otherwise.

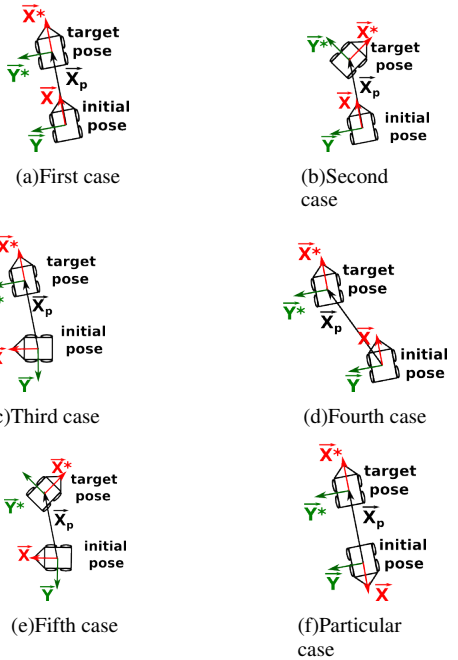
It turns out that five situations may be distinguished as well as a particular case (Fig. 5) :

1. First case (Fig. 5(a)) : a straightforward translation. This occurs when :

$$\overrightarrow{X_p} \equiv \overrightarrow{X} \equiv \overrightarrow{X^*} \tag{6}$$

2. Second case (Fig. 5(b)) : $\overrightarrow{X_p} \not\equiv \overrightarrow{X^*}$ but $\overrightarrow{X} \equiv \overrightarrow{X_p}$.
3. Third case (Fig. 5(c)) : $\overrightarrow{X_p} \not\equiv \overrightarrow{X}$ but $\overrightarrow{X^*} \equiv \overrightarrow{X_p}$.
4. Fourth case (Fig. 5(d)) : $\overrightarrow{X} \equiv \overrightarrow{X^*}$ and $\overrightarrow{X^*} \not\equiv \overrightarrow{X_p}$.
5. Fifth case (Fig. 5(e)) : which corresponds to the general case where $\overrightarrow{X_p} \not\equiv \overrightarrow{X}$, $\overrightarrow{X^*} \not\equiv \overrightarrow{X_p}$ and $\overrightarrow{X} \not\equiv \overrightarrow{X^*}$.

In the following, we discuss the five above-mentioned cases and propose a solution to each of them, gradually.



(a)First case (b)Second case

(c)Third case (d)Fourth case

(e)Fifth case (f)Particular case

Fig. 5 The possible cases to handle the proposed positioning task.

A. First case
In this case (Fig. 5(a)) the three axes $\overrightarrow{X}$, $\overrightarrow{X_p}$ and $\overrightarrow{X^*}$ are coincident. A simple translation using a visual servoing along $\overrightarrow{X}$ can be performed in order to reach the target pose.

B. Second case
In this case (Fig. 5(b)) $\overrightarrow{X} \not\equiv \overrightarrow{X^*}$ but $\overrightarrow{X} \equiv \overrightarrow{X_p}$. In order to satisfy condition of equation (6) the robot needs to perform the following tasks :

- Simulate a target rotation : the visual compass is used once for the computation of the angle θ between the current and the desired images. The desired image is then rotated with this angle as $\mathbf{I}^*_\theta$ to fit the orientation of the current image $\mathbf{I}$. A new target direction $\overrightarrow{X^*_\theta}$ is then obtained.
- Perform a translational visual servoing along $\overrightarrow{X_p}$ using the following control law :

$$\upsilon_X = -\lambda \left(\mathbf{H} + \mu \operatorname{diag}(\mathbf{H})\right)^{-1} \mathbf{L}^{\mathsf{T}}_{\mathbf{I}_\mathcal{S}} \left(\mathbf{I}_\mathcal{S}(\mathbf{r}) - (\mathbf{I}^*_{\mathcal{S}_\theta})(\mathbf{r}^*)\right), \tag{7}$$

  considering $\mathbf{I}^*_{\mathcal{S}_\theta}$ as the spherical representation of $\mathbf{I}^*_\theta$ as described in section 2.1 and $\mathbf{L}_{\mathbf{I}_\mathcal{S}}$ has only one column.
- Perform a robot rotation : regulation of the angle θ to zero (i.e. θ is re-computed for each current image).

$$\omega_Z = \gamma * \theta \tag{8}$$

Where γ is a tunable gain that controls the robot speed to reach the desired orientation.

C. Third case
In this case $\overrightarrow{X}$ and $\overrightarrow{X_p}$ are different but $\overrightarrow{X^*}$ and $\overrightarrow{X_p}$ are coincident. In order to have $\overrightarrow{X_p} \equiv \overrightarrow{X}$ the robot must rotate. A naive approach would be to perform a visual servoing on ω_Z. However, as stated in the introduction (Fig.2(b)), the convergence area of the cost function is limited to 22 degrees. To solve this issue, as in the second case, the desired image is oriented, as $\mathbf{I}^*_\theta$, to fit the current image orientation. The difference wrt the second case is that the axes $\overrightarrow{X}$ and $\overrightarrow{X^*_\theta}$ are only parallel but not coincident. Thus, we consider υ_Y to control the robot orientation. Indeed, υ_Y, computed with visual servoing, is null if and only if $\overrightarrow{X^*_\theta}$ and $\overrightarrow{X}$ are coincident. If not, the sign of υ_Y gives the rotation velocity sign, so :

$$\omega_Z = \alpha * \upsilon_Y, \tag{9}$$

where α is another tunable gain that controls the robot rotation speed to reach the desired orientation.
To sum up, the third case can be solved following these steps :

- simulate a target rotation to obtain $\overrightarrow{X^*_\theta}$ that is parallel to $\overrightarrow{X}$.
- rotate the robot using a visual servoing on υ_Y as described by Equation (9) to make $\overrightarrow{X}$ and $\overrightarrow{X_\theta}$ coincident, for the correct angle, hence coincident with $\overrightarrow{X_p}$ too.
- perform a translational visual servoing along $\overrightarrow{X}$.

D. Fourth case
In this case $\overrightarrow{X} \equiv \overrightarrow{X^*}$ but $\overrightarrow{X} \not\equiv \overrightarrow{X_p}$. Since $\overrightarrow{X^*}$ is already parallel to $\overrightarrow{X}$, we first need to rotate the robot using a visual servoing on υ_Y as in the third case, before performing the pure translation.
Finally, a second pure rotation is performed as the last step of the second case.

E. Fifth case
The axes $\overrightarrow{X}$, $\overrightarrow{X_p}$ and $\overrightarrow{X^*}$ have different orientations. To make visual servoing successful in this general case, we proceed first as the third case and finally considering the last step of the second case. The control algorithm is presented in the next section.

3.2 The proposed solution

Three main stages describe the positioning control algorithm :

1. Step 1 : First pure rotation, based on the initial and the desired images, the robot rotates to point to the desired position (Algorithm 1).

Algorithm 1: First Pure Rotation

Data: Desired image $\mathbf{I}^*$
Result: Velocity component ω_Z.
begin
for *each acquired Image* $\mathbf{I}$ **do**
Find θ the angle between $\mathbf{I}$ and $\mathbf{I}^*$ using a visual compass algorithm.
Rotate the desired image by θ, the result is $\mathbf{I}^*_\theta$.
Perform a visual servoing iteration using $\mathbf{I}^*_\theta$ on υ_Y.
Apply the velocity $\omega_Z = \alpha * \upsilon_Y$ to the robot.
end
end

At the end of the first pure rotation, the robot is not accurately pointing to the desired position. Thus using a pure translation to let the robot reaching the desired position is not sufficient. Hence, robot orientation should simultaneously be controlled with the translation instead of a pure translation control.

2. Step 2 : Combined rotation and translation, the robot is controlled by a simultaneous rotation/translation commands in order to reach the desired position (Algorithm 2).

Algorithm 2: Combined rotation and translation

Data: Desired image $\mathbf{I}^*$
Result: Velocity components υ_X and ω_Z.
begin
for *each acquired image* $\mathbf{I}$ **do**
Find θ the angle between $\mathbf{I}^*$ and $\mathbf{I}$.
Rotate $\mathbf{I}^*$ by θ, the result is $\mathbf{I}^*_\theta$.
Perform a visual servoing iteration using $\mathbf{I}^*_\theta$ on υ_X and υ_Y.
Apply the velocity $(\upsilon_X, \omega_Z = \beta * \upsilon_Y)$ to the robot.
end
end

3. Step 3 : Second pure rotation, finally the robot performs a pure rotation until the angle θ, between the current and the desired image, is null (Algorithm 3).

Algorithm 3: Second pure rotation

Data: Desired Image $\mathbf{I}^*$
Result: Velocity component ω_Z.
begin
for *each acquired image* **I do**
Find angle θ between current image **I** and desired image $\mathbf{I}^*$.
Apply velocity $\omega_Z = \gamma * \theta$ to the robot.
end
end

4. Particular case (Fig. 5(f)) : in this case the axes $\overrightarrow{X} \equiv \overrightarrow{X_p} \equiv \overrightarrow{X^*}$ but $\overrightarrow{X} \equiv -\overrightarrow{X^*}$. In order to accomplish the robot manoeuvre, the robot needs to simulate a target rotation as in the third case so that $\overrightarrow{X_\theta^*}$ has the same orientation as $\overrightarrow{X}$. The robot needs then to perform a backward translation in order to reach the target pose (unlike the previous cases where a forward translation is performed). Finally, the robot accomplishes a pure rotation as the last step of the second case.

3.3 Validation

Experiments were conducted using the setup presented in Fig. 1 : omnidirectional camera and the mobile robot pioneer 3-AT. For these experiments, eight scenarios corresponding to different initial positions on a circle of radius 60 cm (Fig. 6) are carried out. Starting from an initial position on the circle, the robot has to reach the target pose which corresponds to the center of this circle. The arrow corresponds to the robot orientation. We distinguish between two algorithms used in these experiments :

A. Using the proposed classical Omnidirectional Photometric Visual Servoing (classical OPVS)

In these experiments, we use the classical version of omnidirectional photometric visual servoing (*cf.* Section A.). First a desired image is taken and saved in the hard drive of the computer. Then for each of the eight poses, the robot moves using the classical OPVS. The videos showing these experiments can be found in this link : `http://home.mis.u-picardie.fr/~youssef/Videos/OPVS/classical_OPVS/`.

B. Using the enhanced omnidirectional photometric visual servoing

In these experiments, we applied our enhanced version of the omnidirectional photometric visual servoing (enhanced OPVS). It is noteworthy that the enhanced OPVS is also applied for poses 3 and 7 so that we can compare the performance of the enhanced version with the classical one. Regarding the pose 3, the robot is expected to not perform any rotation and then moving backward to the desired pose. Instead, as the video corresponding to the pose 3 shows, the robot performs a rotation of about 180 degrees and moves towards the desired target. This can be explained by the fact the visual compass algorithm outputs a non-zero angle at pose 3 which drives the robot to the next minimum corresponding to $v_Y = 0$ (i.e. particular case of Fig.5-f is almost theoretical). Figure 7 shows the generated paths for poses 1 and 6. The videos corresponding to these experiments can be found in this link `http://home.mis.u-picardie.fr/~youssef/Videos/OPVS/enhanced_OPVS/`.

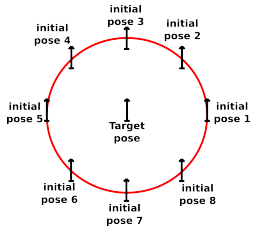
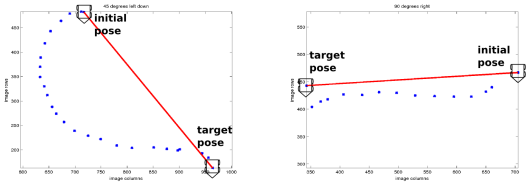


Fig. 6 The experiments with the mobile robot for the positioning task problem.



(a)Pose 6 (b)Pose 1

Fig. 7 Trajectories performed by the robot for the poses 1 and 6. The red line corresponds to the shortest path joining the initial to the target pose and the blue points correspond to the actual path.

3.4 Results analysis

The results in terms of metric error between the final position of the robot and the desired position are measured using a ruler and are presented in Table 1. The error is evaluated for the two schemes corresponding to the classical and enhanced OPVS and according to the X-axis (resp. Y-axis) wich corresponds to ΔX (resp. ΔY).

The parameters α, β, γ are experimentally set as the following $(\alpha, \beta, \gamma) = (5.0, 4.5, 0.5)$. The camera frame rate is 15 fps.

These results show that the enhanced OPVS outperforms the classical OPVS in terms of metric error. Actually, the classical OPVS works only for poses 3 and 7, while the enhanced OPVS works for all the poses. Regarding poses 3 and 7, the classical OPVS provides better results than the enhanced OPVS.

4. Conclusion

In this paper, we presented a new featureless visual servoing algorithm for non-holonomic robot navigation. Considering only image intensities, the proposed visual servoing algorithm allows to control the robot translation and orientation in order to reach the desired pose by performing pure rotation phases and computing the rotation velocity component during the translation step. In future works, we plan to improve the control scheme, still only

OPVS algorithm	Classical OPVS		Enhanced OPVS	
Error(cm) / Poses (Fig. 6)	ΔX	ΔY	ΔX	ΔY
Pose 1	0	60	0	2
Pose 2	2	41	0	5
Pose 3	0	1	4	1
Pose 4	1	47	1	5
Pose 5	5	58	3	3
Pose 6	0	46	0	2
Pose 7	0	1	3	3
Pose 8	4	38	3	0
Mean	1.5	36.5	1.75	2.62
Standard deviation	2	23.17	1.67	1.76

Table 1 Error in centimetres of the enhanced OPVS for positioning task versus the classical OPVS scheme.

considering the photometric feature, but allowing smooth motion instead of a decoupled one.

Acknowledgments

This work was supported by the project COALAS. The project COALAS was selected under the European cross-border cooperation program INTER-REG IV A France (Channel) England, co-funded by the ERDF.

References

[1] H.H. Abdelkader, Y. Mezouar, N. Andreff, and P. Martinet. Image-based control of mobile robot with central catadioptric cameras. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 3522 – 3527, april 2005.

[2] H. Aliakbarpour, O. Tahri, and H. Araujo. Visual servoing of mobile robots using non-central catadioptric cameras. *Robotics and Autonomous Systems*, 62(11) :1613 – 1622, 2014.

[3] J. P. Barreto and H. Araujo. Issues on the geometry of central catadioptric imaging. In *IEEE Int. Conf. on Computer Vision and Pattern Recognition*, volume 2, pages 422–427, Hawaii, USA, December 2001.

[4] G. Caron, E. Marchand, and E. Mouaddib. Photometric visual servoing for omnidirectional cameras. *Autonomous Robots, AURO*, 35(2) :177–193, October 2013.

[5] F. Chaumette and S. Hutchinson. Visual servo control, Part I : Basic approaches. *IEEE Robotics and Automation Magazine*, 13(4) :82–90, December 2006.

[6] Yoshihiko Mochizuki and Atsushi Imiya. Featureless visual navigation using optical flow of omnidirectional image sequence. In *Workshop of SIMPAR*, volume 8, pages 307–318, 2008.

[7] D. Scaramuzza and R. Siegwart. Appearance-guided monocular omnidirectional visual odometry for outdoor ground vehicles. *IEEE Trans. on Robotics*, 24(5) :1015–1026, Oct 2008.

[8] R. Tatsambon Fomena, H. Yoon, A. Cherubini, F. Chaumette, and S. Hutchinson. Coarsely calibrated visual servoing of a mobile robot using a catadioptric vision system. In *IEEE Int. Conf. on Intelligent Robots and Systems*, pages 5432–5437, St Louis, USA, October 2009.

[9] Jianguo Zhao, Yunyi Jia, Ning Xi, Weixian Li, Bo Song, and Liang Sun. Visual servoing using non-vector space control theory. In *Advanced Intelligent Mechatronics (AIM), 2012 IEEE/ASME International Conference on*, pages 87–92. IEEE, 2012.

Omnidirectional photometric visual path following for wheelchair autonomous driving

Youssef Alj, Guillaume Caron, Nicolas Ragot

Omnidirectional photometric visual path following for wheelchair autonomous driving

Youssef Alj
Université de Picardie Jules Verne
MIS Laboratory
33 rue Saint-Leu,
80039 Amiens Cedex, France
Email: youssef.alj@u-picardie.fr

Guillaume Caron
Université de Picardie Jules Verne
MIS Laboratory
33 rue Saint-Leu,
80039 Amiens Cedex, France
Email: guillaume.caron@u-picardie.fr

Nicolas Ragot
ESIGELEC, IRSEEM laboratory
Technopôle du Madrillet,
Avenue Galilée, BP 10024
Saint Etienne du Rouvray
CEDEX, France
Email: nicolas.ragot@esigelec.fr

***Abstract*—In this paper we address the issue of autonomous wheelchair navigation. Using an omnidirectional camera, we propose a system that allows wheelchair driving using a visual path following technique. First, a visual path composed of target images is acquired. Second, successive visual servoings are performed by minimizing the error between the current and a target image. Experiments were conducted on a variety of paths: straight line, curved line and forward/backward movements. The ground truth is obtained using a Vicon system that records wheelchair trajectories during the learning and the path following steps.**

I. INTRODUCTION

Since the last few years, European and North-American public policies are facing major issues related to people with disabilities. Several studies highlight the increasing demand for social care and a steadily growing number of wheelchair users [1]. The cost of caring for people with disabilities in European countries has been recently estimated to 795 billion Euros [2]. Within this framework, the national authorities are moving towards innovative solutions based on robotics and ICT for caring for the user, enhancing his independence and his quality of life.

Within this framework, the European project INTERREG IVA COALAS (Cognitive Assisted Living Ambient System) started in November 2012 and finishing in 2015 aims to bring solutions to these issues. This project is focusing on designing and developing an autonomous cognitive platform combining an intelligent wheelchair (equipped with a set of heterogeneous sensors) and assistive capabilities of a humanoid robot. The project is structured in a user-centred co-creation process aiming to give responses to new needs and uses. Three milestones are defining the project framework:

- expectations and recommendations
- design and development
- evaluation

The work presented in this paper addresses the "design and development" action, particularly the vision-based wheelchair (Figure 1) autonomous navigation. Several studies related to this research topic have recently been carried out: assistive semi-autonomous or autonomous navigation of powered wheelchair. These works handle mainly the problems of obstacle avoidance and doorway passing [3], [4] based on ultrasound sensor dataset. Other studies are related to assistive autonomous navigation based on LIDAR or Kinect sensors [5], [6], [7], [8], [9]. Indeed, while range sensors almost directly bring a distance from a measure, these active sensors spend more energy than passive ones. In some medical contexts, the emission of ultrasounds, laser or IR light is not allowed. A passive camera easily avoids theses issues and is not range limited as range sensors are. Furthermore, using a passive camera can be used for another vision-based tasks such as person recognition. However, computing distances from images may be tedious if reliable measures are expected, and this is the core issue tackled by many computer vision works. For this reason, we propose to implement a photometric based wheelchair navigation.

The rest of the paper is organized as follows. Studies related to vision-based wheelchair navigation are first presented. Then, the implementation method is described and experiments are presented before the conclusion.

Figure 1: The wheelchair equipped with the omnidirectional camera, on top of it surrounded by the red ellipse.

II. RELATED WORKS

Vision-based wheelchair autonomous or semi-autonomous navigation has recently received attention from researchers.

Semi-autonomous navigation has been tackled using a camera, oriented forward, to assist the wheelchair driving in order to constrain its lateral position in a corridor [10]. In this work, straight line image features are extracted in order

to compute a vanishing point that is used in a control loop of which the task is to have the latter vanishing point at the center of the image.

The latter kind of navigation, even interactive, is reactive and is not sufficient to implement the navigation from a known point A to a known point B. In order to implement such functionality, we can use visual paths. A visual path is defined as a set of images acquired by a camera embedded on a mobile unit during its motion. Indeed, in a general way, the goal of a mobile robot or a wheelchair is not visible from its initial or current position. That is why a set of waypoints is necessary to ensure a guidance to the goal. Once the visual path is known, the autonomous navigation must move the wheelchair to reach every image of the path, successively. While some works deal with 3D reconstruction and mapping along the visual path to increase the amount of available information needed during navigation [11], [12], others directly deal with image intensities [13]. In the latter work, the mutual information shared by the current and the desired images is exploited to build the autonomous vehicle control law that maximizes the mutual information. Thus, waypoints are implicitly defined by images only, composing a photometric visual path. Each waypoint is only linked to the previous and next images along the path so [13] is not a metric navigation but a topological one [14]. The advantage is that this method needs a lighter memory and a lower processing cost when planning the path. Actually, only the links between known places are considered. These latter works were developed for autonomous vehicles context in which a wheelchair may be seen as an instance.

Generally speaking, a perspective camera is used for the vision-based wheelchair control. The perceived field may be limited such that its occlusions are a big issue when comparing the current image to the visual memory. Despite installing additional cameras on the wheelchair is a theoretically valid solution, it increases costs, takes room and introduces synchronization and photometric inconsistency issues between cameras that have to be tackled. The omnidirectional vision brings an elegant answer to these issues, allowing to acquire a panoramic image with a single camera and a catadioptric optics made of a convex mirror and a set of lenses. Even if an extension of 3D reconstruction based visual path following was proposed for wide field of view cameras [15], it needs precise camera calibration and a lot of preprocessing (feature detection, matching or tracking, 3D reconstruction) before producing a control to drive the robot. By making perception and action a bit closer, [16] proposes to address the wheelchair visual path following as a set of successive visual homing, only exploiting 2D features that are based on SIFT features and 2D camera motion estimation. Based on the same idea of executing successive visual homing on each image of the visual path, [17] introduces the omnidirectional photometric visual servoing to control a mobile robot over a visual path. The latter work avoids any feature detection and matching issues (e.g. motion estimation) in order to compute the robot control inputs. The control inputs are obtained from the difference between the current image and the image to be reached. We therefore propose to adapt this work to the autonomous wheelchair navigation issue.

III. THE PROPOSED OMNIDIRECTIONAL PHOTOMETRIC VISUAL PATH FOLLOWING

The omnidirectional photometric visual path following is implemented as a set of visual servoings successively applied on the list of waypoint images defining the visual path. Image-based visual servoing is the closed-loop control of a robotic system of which the task is defined in the image [18].

In the more precise case of wheelchair control, since the one considered in the experiments of this paper has two independently motorized wheels (back left and right of the wheelchair) and two free wheels, we consider it follows a unicycle model. Such a model only involves two degrees of freedom, that are longitudinal and rotational velocities. Furthermore, in order to have a panoramic visual perception around the wheelchair, the camera axis must be vertical, and by installing the omnidirectional camera at well chosen position, its optical axis and the wheelchair rotation axis are aligned, as well as the X-axis of the camera and the longitudinal axis of the wheelchair. Thus, we can consider the extrinsic calibration between the wheelchair and the camera as being the identity matrix. Experiments shown in Section IV validate the robustness of the visual servoing to this approximation.

In this paragraph, the proposed software architecture is presented. We use ROS (Robot Opœerating System [19]) for our software development methodology. One of the major advantages of using ROS, is the ability to develop algorithms regardless of the used hardware. Figure 2 illustrates this feature. The visual servoing node subscribes to the uEye camera topic `/camera/image_color` which delivers the captured images. Using these images, the visual servoing node generates velocity messages and publishes them to different kinds of hardware platforms (i.e. pioneer 3AT robot, electrical powered wheelchair, etc.). Thanks to this ROS-based software architecture, one can only change the robot topic name (`/RosAria/cmd_vel` or `/Wheelchair/cmd_vel`) to get the algorithms to work.

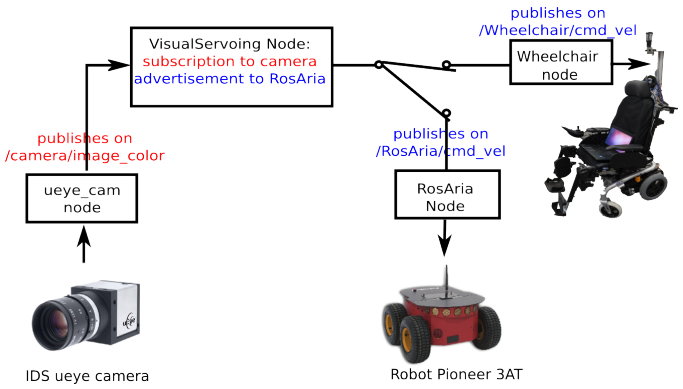
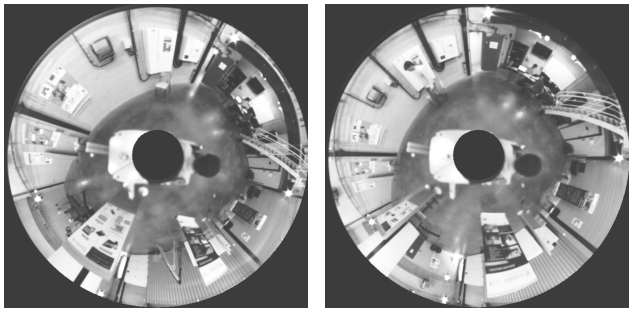


Figure 2: ROS based software architecture.

IV. EXPERIMENTS

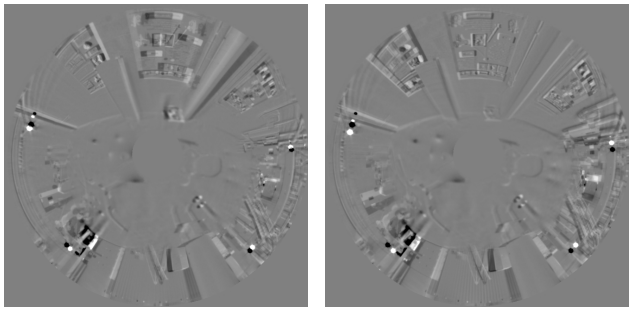
We performed our experiments on three kinds of paths: straight line path, curved line path, and forward-backward movements.

The first step of each experiment is the acquisition of the visual path. For all the experiments presented in this article, 1 frame every 50 is captured for making the so-called visual path.



(a) (b)

Figure 3: Some images of the learned path.



(a) (b)

Figure 4: Difference images during visual servoing.

This number is chosen so that the initial and target images overlap. The camera frame-rate is about 15 FPS. The motion of the wheelchair is controlled using the wheelchair's joystick provided by the wheelchair manufacturer or by a remote one. The set of captured frames are stored in the hard drive, each of these images is considered as a target image.

In the second step, these captured images are loaded and the visual path is followed using successive visual servoings. Starting from an initial position, the wheelchair moves until the photometric error between the current and the target image is below a certain threshold. This threshold is set empirically and may vary depending on the texture richness of the surrounding environment. Once the photometric error is below this threshold, the following target image is considered and the error is minimized. Figure 3 shows some images of a visual path for the curved trajectory presented in section IV-A. Figure 4 shows the image difference between the current and the target image.

Camera calibration was performed using the approach developped by [20]. The time needed to process each frame is about 60ms.

Experiments were carried out in a dedicated room equipped with a 20-camera VICON system (precision less than the millimeter) to record the learned and followed trajectories. Once trajectories are recorded, error between the followed and the learned paths is computed: for each point (x_f, y_f) in the followed path, we find the nearest point (x_{nl}, y_{nl}) in the learned path and compute the error between the two points as the following:

$$\text{error}_{(x_f,y_f)} = \sqrt{(x_f - x_{nl})^2 + (y_f - y_{nl})^2} \qquad (1)$$

Note that Vicon points and images captured by the omnidirectional camera are not synchronized. Thus points generated by the Vicon do not correspond necessarily to the acquired images.

A. *Curved line trajectory*

In this section we present results regarding a curved line trajectory. Results are shown in Figure 5 and in Figure 6. In this experiment, 16 images are considered to form the learning path. As in the straight line trajectory case, the error between the two paths is minimal as far as the wheelchair approaches to a target image.

However, in the case of curved line trajectory, the error in getting to be large as far as the wheelchair approaches the final target position.

The Video of the learned trajectory can be found in http://home.mis.u-picardie.fr/~youssef/Videos/curved_path/curved_learned_path.mpg. The video showing the followed trajectory can be found here: http://home.mis.u-picardie.fr/~youssef/Videos/curved_path/curved_path_visual_servo.mpg

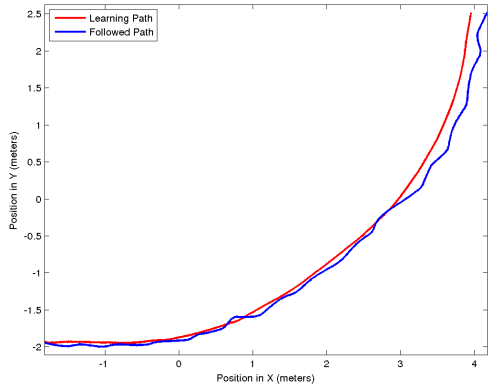


Figure 5: Followed and learned paths for a curved line trajectory.

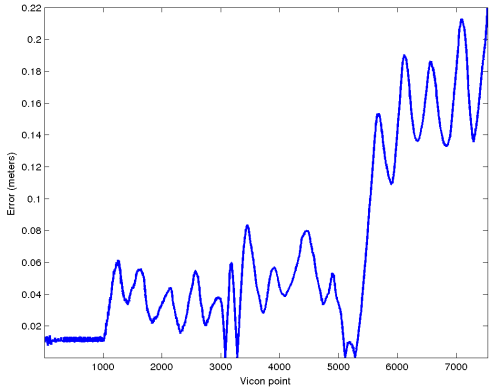


Figure 6: Error between learned and followed paths for a curved line trajectory.

B. Results synthesis and discussion

Table I summarizes the results presented above. According to this table, the error between the followed and the learned path in the case of a straight line trajectory is smaller than in the other scenarios: the mean error between the two paths in the case of a straight line is only 2 centimetres while it reaches 17 centimetres in the case of forward/backward movements.

When the threshold error is too large, the wheelchair performs several rotations during its trajectory and our system is not able to correct such rotation in order to reach the target position. On the contrary, if the threshold error is too small, several iterations are needed in order to reach the minimum error.

It turns out that our approach is best suited for straight line paths. We plan to investigate how to improve the proposed solution in order to correct rotation during visual servoing.

In order to avoid manual threshold setting, we also plan to study how we can model the photometric error evolution with a 2-degree polynomial. This allows us to move to the next target image once the error stops decreasing

Trajectories	Straight	Curved	Forward/Backward
min (meters)	1.96e-04	1.75e-04	4.15e-09
max (meters)	0.07	0.22	0.51
mean (meters)	0.02	0.06	0.17
standard deviation (meters)	0.01	0.05	0.12

Table I: Error statistics (in meters) between the followed and learned path for different trajectories.

V. CONCLUSION

In this paper we presented a powered wheelchair navigation algorithm based on an omnidirectional camera and visual servoing technique. We proposed several trajectories to validate our experiments. These experiments have shown that our approach is best suited for straight paths and for slightly curved paths. In the future work, we will focus on how to improve the proposed approach to handle paths with higher curvature. We also plan to investigate how to model the error using a 2-degree polynomial which avoids manual tuning of the error threshold. Finally, as in the approach proposed by [21] we want to address the issue of merging manual and robot control for determining the appropriate blending for wheelchair navigation.

ACKNOWLEDGMENT

This work has been supported by the COALAS project. The COALAS project has been selected in the context of the INTERREG IVA France (channel) - England European Cross-border Co-operation Program, which is co-financed by the ERDF. Authors thank Yassine Nasri and Vincent Vauchey from ESIGELEC for their valuable work in the wheelchair electronic and mechanical equipment.

REFERENCES

[1] E. Torta, J. Oberzaucher, F. Werner, R. H. Cuijpers, and J. F. Juola, "Attitudes towards socially assistive robots in intelligent homes: Results from laboratory studies and field trials," *Journal of Human-Robot Interaction*, vol. 1(2), pp. 76–99, 2012.

[2] A. Gustavsson, M. Svensson, and F. Jacobi, "Cost of disorders of the brain in europe 2010," *European neuropsychopharmacology: the journal of the European College of Neuropsychopharmacology*, vol. 21, pp. 718–779, 2011.

[3] S. Wang, L. Chen, H. Hu, and K. McDonald-Maier, "Doorway passing of an intelligent wheelchair by dynamically generating bézier curve trajectory," in *Int. Conf. on Robotics and Biomimetics*, Guangzhou, China, December 11-14 2012, pp. 1206–1211.

[4] M. Gillham, G. Howells, S. Kelly, S. Spurgeon, and M. Pepper, "Real-time doorway detection and alignment determination for improved trajectory generation in assistive mobile robotic wheelchairs," in *Int. Conf. on Emerging Security Technologies, EST*, Cambridge, UK, September 9-11 2013.

[5] D. Vanhooydonck, E. Demeester, A. Hintemann, J. Philips, G. Vanacker, H. Van Brussel, and M. Nuttin, "Adaptable navigational assistance for intelligent wheelchairs by means of an implicit personalized user model," *Robotics and Autonomous Systems, ROBOT*, vol. 58, pp. 963–977, 2010.

[6] M. Tomari, Y. Kobayashi, and Y. Kuno, "Development of smart wheelchair system for a user with severe motor impairment," in *Int. Symp. on Robotics and Intelligent Sensors*, vol. 41, 2012, pp. 538–546.

[7] R. Li, M. Oskoei, and H. Hu, "Towards ros based multi-robot architecture for ambient assisted living," in *IEEE Int. Conf. on Systems, Man and Cybernetics*, Manchester, U.K, October 13-16 2013, pp. 3458–3463.

[8] T. Theodoridis, H. Hu, K. McDonald-Maier, and D. Gu, "Kinect enabled monte carlo localisation for a robotic wheelchair," in *Int. Conf. on Intelligent Autonomous Systems*, Jeju Island, Korea,, June 26-29 2012.

[9] M. Derry and B. Argall, "Automated doorway detection for assistive shared-control wheelchairs," in *IEEE Int. Conf. on Robotics and Automation*, May 2013, pp. 1254–1259.

[10] F. Pasteau, A. Krupa, and M. Babel, "Vision-based assistance for wheelchair navigation along corridors," in *IEEE Int. Conf. on Robotics and Automation, ICRA*, Hong-Kong, Hong-Kong, Jun. 2014.

[11] E. Royer, M. Lhuillier, M. Dhome, and J. Lavest, "Monocular vision for mobile robot localization and autonomous navigation," *Int. Journal of Computer Vision*, vol. 74, no. 3, pp. 237–260, 2007.

[12] M. Meilland, P. Rives, and I. Comport, A., "Dense RGB-D mapping of large scale environments for real-time localisation and autonomous navigation," in *Intelligent Vehicle (IV'12) Workshop on Navigation, Perception, Accurate Positioning and Mapping for Intelligent Vehicles*, Alcala de Henares, Spain, Jun. 2012.

[13] A. Dame and E. Marchand, "Using mutual information for appearance-based visual path following," *Robotics and Autonomous Systems, ROBOT*, vol. 61, no. 3, pp. 259–270, March 2013.

[14] O. Trullier and J. Meyer, "Biomimetic navigation models and strategies in animats," *AI Commun.*, vol. 10, no. 2, pp. 79–92, Apr. 1997.

[15] J. Courbon, Y. Mezouar, and P. Martinet, "Autonomous navigation of vehicles from a visual memory using a generic camera model," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 3, pp. 392–402, Sept 2009.

[16] T. Goedeme, T. Tuytelaars, L. Van Gool, G. Vanacker, and M. Nuttin, "Feature based omnidirectional sparse visual path following," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, IROS*, Aug 2005, pp. 1806–1811.

[17] G. Caron, E. Marchand, and E. Mouaddib, "Photometric visual servoing for omnidirectional cameras," *Autonomous Robots, AURO*, vol. 35, no. 2, pp. 177–193, October 2013.

[18] F. Chaumette and S. Hutchinson, "Visual servo control, Part I: Basic approaches," *IEEE Robotics and Automation Magazine*, vol. 13, no. 4, pp. 82–90, December 2006.

[19] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2, 2009, p. 5.

[20] G. Caron and D. Eynard, "Multiple camera types simultaneous stereo calibration," in *IEEE Int. Conf. on Robotics and Automation, ICRA*, Shanghai, China, May 2011, pp. 2933–2938.

[21] A. Goil, M. Derry, and B. Argall, "Using machine learning to blend human and robot controls for assisted wheelchair navigation." in *IEEE Int. Conf. on Rehabilitation Robotics*, 2013, pp. 1–6.

Modélisation de Scènes Naturelles à Partir de Séquences Vidéos Multi-vue plus Profondeur (MVD)

Youssef Alj[1,2] Guillaume Boisson[1] Philippe Bordes[1] Muriel Pressigout[2] Luce Morin[2]

[1] Technicolor
[2] INSA Rennes

{youssef.alj,guillaume.boisson,philippe.bordes}@technicolor.com
{muriel.pressigout,luce.morin}@insa-rennes.fr

Résumé

Dans cet article, un schéma de modélisation de séquences Multi-vues Vidéo plus profondeur (MVD) est présenté. Le but est de réduire la redondance de profondeur et de texture présentes dans les séquences MVD. Pour ce faire, la fusion de cartes de profondeurs utilisant une représentation volumétrique est proposée. Les voxels sont "carvés" itérativement pour chaque vue en utilisant la technique de traçage de rayons (ray tracing). La surface fusionnée est extraite à partir de cette représentation en utilisant l'algorithme de Marching Cubes. Finalement, le problème de plaquage des textures sur cette surface résultante est abordé. L'algorithme proposé sélectionne parmi toutes les textures le meilleur candidat pour texturer un triangle de la surface résultante. Ce choix est fait en utilisant une métrique dite de photocohérence. Les tests et les résultats sont fournis pour des images fixes en utilisant les séquences MVD usuelles.

Mots clefs

Multi-view Video plus Depth (MVD), 3DTV, FTV, depth map fusion, Space Carving, multi-view texture mapping.

1 Introduction

L'Imagerie Multi-Vue (IMV) a suscité un vif intérêt durant les dernières décennies. Grâce au développement des écrans stéréoscopiques, l'IMV fournit une sensation réaliste de profondeur à l'utilisateur et une navigation virtuelle autour de la scène observée, ouvrant par conséquent un large éventail de sujets de recherche et d'applications telles que la télévision 3D (TV3D) ou la FTV (*Free-viewpoint TV*). Cependant, de nombreux défis techniques ont entravé la rapide apparition de ces applications dans les marchés de masse. Ces défis peuvent être liés à l'acquisition de la scène et à sa représentation d'une part ou à la transmission des données représentées d'autre part. Dans le contexte de la représentation de scènes naturelles, de nombreux efforts ont été fournis afin de surmonter ces difficultés. Les méthodes proposées dans la littérature peuvent être classées en trois catégories : les représentations basées image, les représentations basées géométrie ou les représentations intermédiaires.

Les représentations basées image regroupent l'utilisation de MVV (Multi-view Video), 2D+Z [1] ou MVD (*Multi-view Video plus Depth*) [2], où chaque vue est constituée de l'image acquise et d'une carte de profondeur estimée. Les représentations basées géométrie s'appuient sur l'utilisation d'un modèle géométrique à base de surface maillée [3], de modèle volumétrique [4] ou de nuage de points [5]. Les représentations intermédiaires incluent l'utilisation des LDI (*Layered Depth Images*) [6], des représentations dites *billboards* [7] ou de soupe de polygones [8]. L'approche adoptée dans cet article consiste en une méthode hybride s'appuyant sur l'utilisation des séquences MVD, afin de conserver le photo-réalisme de la scène observée, combinée avec un modèle géométrique, à base de maillage triangulaire, renforçant ainsi la compacité de la représentation.

L'utilisation des représentations volumétriques pour la modélisation de scènes a connu un grand succès ces dernières années grâce à la robustesse de ces méthodes aux différents bruits. Typiquement, ces méthodes supposent l'existence d'un volume englobant dans lequel la scène d'intérêt se situe. Ce volume est ensuite subdivisé en éléments cubiques appelés voxels. Durant l'étape de sculpture de l'espace (*Space Carving*), chaque voxel est étiqueté comme étant opaque ou transparent selon sa cohérence avec la scène. Ceux qui sont cohérents sont déclarés comme opaques et les autres voxels sont étiquetés comme transparents. La surface finale est obtenue à partir de cette classification binaire en utilisant l'algorithme de *Marching Cubes*. La mesure de la cohérence d'un voxel avec la scène peut être basée sur l'information de silhouette extraite des images d'entrées, cette méthode est connue sous le nom de *Shape from silhouette* (modélisation à partir des silhouettes) [9] ou directement à partir de l'information contenue dans les images d'entrées ce qui est connue sous le nom de *Shape from photoconsistency* (modélisation à partir de la photocohérence) [10]. Les approches alternatives comme VRIP [11] ou KinectFusion [12] utilisent les cartes de profondeur comme données d'entrée et les fusionnent en discrétisant une fonction de distance signée sur le volume englobant,

la valeur de cette distance en chaque voxel correspond à la distance signée du voxel à la plus proche surface vis à-vis du point de vue. Dans VRIP, une surface explicite est extraite à partir de cette représentation volumétrique grâce l'algorithme de Marching Cubes, tandis que, contraints par le rendu en temps réel, cette surface est simplement prédite par la technique de traçage de rayons dans le cas de KinectFusion. La méthode proposée dans cet article est similaire à ces dernières approches utilisant les cartes de profondeur. Néanmoins, l'objectif recherché est de construire un modèle 3D explicite destiné à transmettre l'information contenue dans les cartes de profondeur originales en réduisant les redondances. La cohérence du modèle 3D avec les données d'entrée est donc le critère de qualité qui nous intéresse dans cette étude. En particulier, ce modèle doit être photocohérent avec les images d'entrée, la photocohérence est donc renforcée grâce à l'algorithme de plaquage de texture proposé. Les contributions de cet article sont doubles : d'abord, un schéma volumétrique dédié à la fusion des cartes de profondeur en une surface maillée est proposé, c'est l'objet de la section 2. Ensuite, un nouvel algorithme de plaquage texture multi-vues est présenté dans la section 3. Enfin, les résultats sont présentés dans la section 4.

2 Modélisation géométrique

Dans cette section, notre schéma volumétrique dédié à la fusion des cartes de profondeur est présenté. Considérant le cas général, les séquences d'entrées sont calibrées mais ne sont pas nécessairement alignées ni rectifiées. Le problème d'estimation de la disparité inter-vue n'est pas abordé dans cet article et les cartes de profondeurs fournies sont supposées être fiables. Le but est donc de fusionner ces cartes de profondeur en une seule surface cohérente avec les vues originales. Une vue d'ensemble de notre schéma de fusion est présentée dans la figure 1. Premièrement, un maillage triangulaire en haute résolution est extrait à partir de chaque carte de profondeur. Ensuite, le volume englobant tous ces maillages est défini puis discrétisé en voxels. Un voxel peut avoir deux statuts : opaque ou transparent. Pour chaque caméra, l'ensemble des voxels se situant dans son cône de vue sont déterminés et étiquetés comme opaque. A l'étape suivante, l'Enveloppe Volumétrique du Maillage (EVM) est déterminée pour chacune des vues. Étant donné un maillage correspondant à un point de vue, l'EVM définit l'ensemble des voxels situés sur la surface maillée. Notre formulation de l'algorithme de *Space Carving* repose sur l'EVM calculée. En effet, pour chaque vue, des rayons sont tracés à partir du centre de la caméra à chaque voxel constituant l'EVM correspondant à cette vue. Les voxels se situant sur chaque rayon sont déclarés transparents.

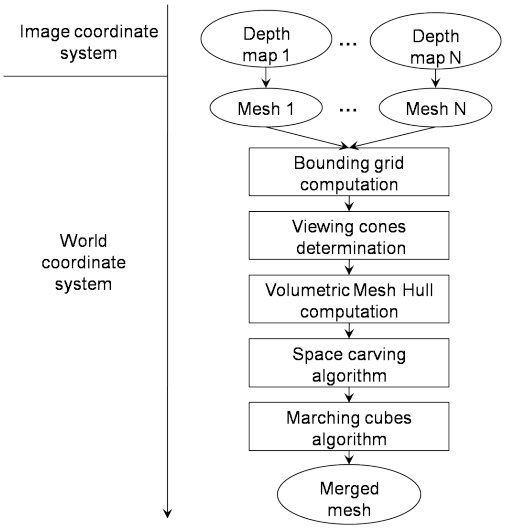


Figure 1 – *Structure du schéma de fusion.*

2.1 Construction des maillages à partir des cartes de profondeur

La première étape de l'algorithme est la génération d'un maillage pour chaque carte de profondeur donnée. Pour chaque vue, un maillage haute résolution est construit dans le repère image. Chaque sommet du maillage représente un pixel dans la carte de profondeur. Les maillages ainsi construits sont alignés en exprimant chaque sommet du maillage dans le repère monde en utilisant l'équation suivante :

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{pmatrix} & R & & T \\ & & & \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} f_x & S_{uv} & c_u & 0 \\ 0 & f_y & c_v & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}^{-1} \begin{pmatrix} su \\ sv \\ s \\ 1 \end{pmatrix}$$

Où X, Y, et Z représentent les coordonnées d'un point 3D dans le repère monde. u et v sont les coordonnées de la projection du point 3D dans le repère image, et $s = z_{cam}$ est la profondeur du pixel (u, v) dans le repère caméra. R et T définissent respectivement la matrice de rotation et de translation du repère caméra par rapport au repère monde. f_x, f_y, S_{uv}, c_u et c_v sont les paramètres intrinsèques de la caméra.

2.2 Calcul de la grille englobante

Les maillages ainsi construits sont les entrées de notre représentation volumétrique. La structure de données utilisée dans cette représentation est une grille volumétrique. Cette grille est construite par subdivision régulière de la boîte englobante de l'ensemble des maillages en éléments parallélépipédiques appelés voxels. Un étiquetage binaire est utilisé pour attribuer à chaque voxel l'étiquette opaque ou transparent. L'étiquette de chaque voxel est initialisée à transparent. Les étiquettes des voxels sont modifiées tout

au long de l'algorithme selon leur pertinence pour représenter la scène.

2.3 Détermination des cônes de vue

Dans cette étape le but est de trouver l'ensemble des voxels se situant dans le cône de vue de chaque caméra. Les voxels en dehors de l'union des cônes de vue sont étiquetés comme transparent. Le cône de vue de chaque caméra est modélisé par un frustum défini par quatre plans délimitant l'ensemble des sommets visibles selon chaque axe (voir Figure 2). Le but est donc de déterminer pour chaque caméra l'ensemble des voxels se situant dans le frustum correspondant. Les équations des plans du frustum sont donc calculés pour chaque caméra. Ces équations nous permettent de situer chaque voxel par rapport aux plans calculés et donc de définir les voxels dans chaque frustum.

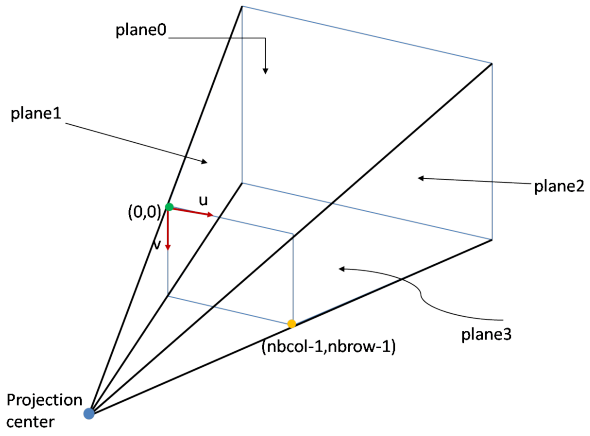


Figure 2 – *Calcul des plans du Frustum de projection.*

2.4 Construction de l'Enveloppe Volumétrique du Maillage EVM

Dans cette section, le calcul de l'Enveloppe Volumétrique du Maillage (EVM) est présenté. Cette EVM sera utilisée durant le processus de *Space Carving*. Pour chaque maillage construit, l'ensemble des voxels intersectant la surface du maillage définit l'EVM associée à ce maillage. A la fin de la construction de chaque EVM, chaque voxel aura une information additionnelle définissant à quelle EVM ce voxel appartient. Pour chaque maillage, les triangles sont parcourus et analysés selon la grille volumétrique en utilisant un algorithme de *Scan Line 3D*, cette analyse permet de définir les voxels se situant sur chaque triangle. L'ensemble des voxels analysés seront marqués comme appartenant à l'EVM courante en mettant à jour l'information additionnelle associée au voxel. L'algorithme d'analyse des triangles d'un maillage est présenté dans l'algorithme 1.

```
// Parcours des triangles
pour chaque triangle T faire
    pour chaque axe X, Y et Z faire
        Calculer les limites entières b_min et b_max de T
        selon chaque axe.
        pour m ← b_min to b_max faire
            Calculer l'intersection du triangle T avec le
            plan à la coordonnée m.
            Trouver les voxels au long de cette
            intersection en utilisant la ligne de Bresenham.
            Ajouter ces voxels à l'EVM.
        fin
    fin
fin
```

Algorithm 1: Détermination de l'Enveloppe Volumétrique du Maillage (EVM).

2.5 Algorithme de *Space Carving*

Jusqu'à cette étape de l'algorithme, un voxel opaque est un voxel qui se situe dans au moins un cône de vue et un voxel de l'EVM est un voxel qui se situe sur un des maillages. Dans cette étape, le *Space Carving* est effectué en mettant à jour les étiquettes des voxels d'opaque à transparent pour tous les voxels situés devant l'EVM. Plus précisément, les différents EVMs sont considérés successivement et un critère de cohérence géométrique est utilisé afin de mettre à jour les étiquettes des voxels. Ce critère est basé sur la position relative du voxel par rapport au maillage considéré et le point de vue correspondant à ce maillage. Un voxel est dit géométriquement cohérent avec une surface du maillage s'il se situe derrière cette surface par rapport au point de vue correspondant. Les voxels détectés comme géométriquement incohérents sont étiquetés comme transparents. Le processus de *Space Carving* met à jour le volume sculpté pour chaque caméra. Ce volume est initialisé à l'union des voxels situés dans les cônes de vues calculés dans la section 2.3 (voir Figure 3). Le même volume est ensuite raffiné itérativement avec l'information géométrique provenant de chacune des vues, voir Figures 3 et 4), en sculptant les zones occultantes relatives à chaque vue. Pour chaque vue, les rayons sont tracés du centre de la caméra aux voxels de l'EVM correspondant à cette caméra. Pour chaque rayon, les voxels se situant sur le rayon courant sont scannés en utilisant l'algorithme de Bresenham 3D de traçage de lignes. Comme ils sont géométriquement incohérents, ces voxels seront creusés (i.e. étiquetés comme transparents). A la fin de cette étape de *Space Carving*, l'ensemble des voxels opaques définit le modèle volumétrique de nos données MVD.

2.6 Marching Cubes

Finalement, la surface fusionnée est extraite à partir du volume binairement étiqueté en utilisant l'algorithme des

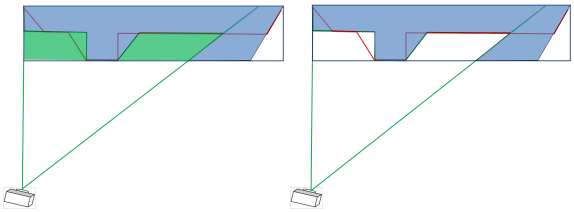


Figure 3 – *Space carving vis-à-vis de la caméra gauche. Les rayons sont tracés du centre de la caméra à l'EVM correspondante. Les zones vertes sont carvées (schéma à gauche). Le résultat du Space Carving vis-à-vis de la caméra gauche est montré dans le schéma à droite.*

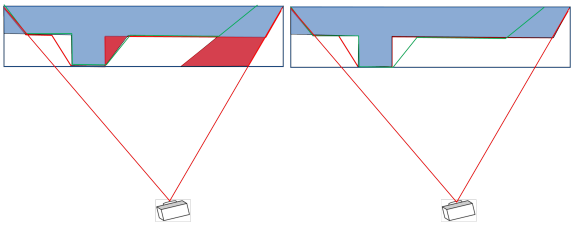


Figure 4 – *Space carving pour la caméra droite. Les zones rouges sont carvées. Le résultat est montré dans le schéma de droite.*

Marching Cubes. L'algorithme de *Marching Cubes* a été introduit par Lorenson et Cline [13]. Cet algorithme prend comme entrée une structure de données de type volume régulièrement subdivisé en voxels. A chaque voxel de ce volume est attribué une valeur. Durant le traitement, chaque sommet de voxel ayant une valeur supérieure ou inférieure à une valeur prédéfinie est marqué. Tous les autres sommets restent non-marqués. Par conséquent, le résultat de l'algorithme de *Marching Cubes* est le maillage triangulaire connectant tous les sommets qui ont été marqués.

3 Plaquage des textures

La contrainte de photocohérence est renforcée en texturant la surface fusionnée extraite de l'algorithme de *Marching Cubes*. Les images de textures des séquences MVD sont

```
// Parcours des vues
pour chaque caméra faire
    Extraire la position de la caméra.
    Extraire l'EVM correspondante.
    pour chaque voxel dans l'EVM faire
        Trouver la ligne de Bresenham 3D entre la
        position de la caméra et le centre du voxel.
        Mettre à jour les labels des voxels se situant sur
        cette ligne à transparent.
    fin
fin
```

Algorithm 2: L'algorithme de *Space Carving*.

utilisées afin de texturer cette surface et un nouvel algorithme de plaquage de textures multi-vues basé sur une métrique de photocohérence est proposé. Les principales étapes de cet algorithme sont présentées dans la Figure 5. D'abord, la visibilité est déterminée pour chaque triangle. Ensuite, chaque texture est projetée sur les triangles visibles de cette surface. Les erreurs de projections photométriques sont calculées pour chaque triangle et pour chaque vue et sont sauvegardées dans des images d'erreurs. La meilleure texture pour chaque triangle est celle qui minimise une métrique de photocohérence. Les triangles qui ne sont visibles par aucune caméra sont texturés avec la texture de la caméra intermédiaire. Les vues synthétisées sont extraites en faisant le rendu du maillage texturé, chaque triangle étant texturé avec la meilleure texture selon la métrique de photocohérence.

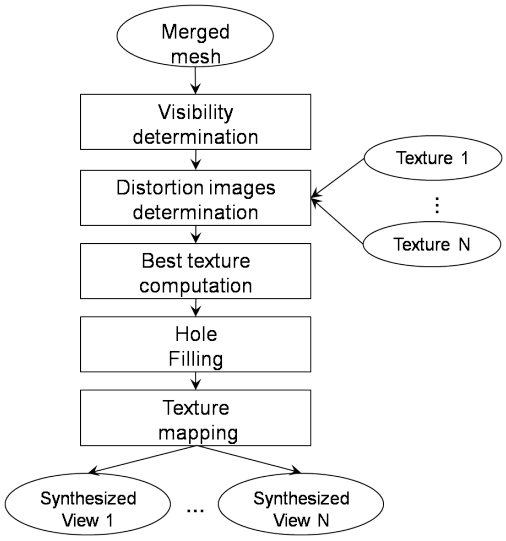


Figure 5 – *Le schéma global de notre algorithme de plaquage de texture basé sur la métrique de photocohérence.*

A partir de l'ensemble des images de texture $\mathcal{I} = \{I_1, \ldots, I_n\}$ et le maillage $\mathcal{M}$, le but est de trouver pour chaque triangle $T \in \mathcal{M}$ la meilleure image de texure $\hat{I}_T \in \mathcal{I}$. Texturer un triangle T est formulé comme un problème de minimisation d'énergie. La meilleure texture pour T est calculée en minimisant une fonction de coût décrivant la photocohérence du triangle T avec l'ensemble des images de texture. La photocohérence est estimée en projetant le maillage texturé sur les vues $\mathcal{V} = \{V_1, \ldots, V_n\}$ correspondant au images d'entrées $\mathcal{I}$. Le triangle t (Voir Figure 6) est d'abord texturé avec la texture I_1 de la caméra 1 (flèche rouge pour l'opération de plaquage de texture) puis projeté sur les caméras 3, 4 et 5, i.e. les caméras dans lesquelles le triangle t est visible (flèche verte pour l'opération de projection). Les erreurs quadratiques entre le triangle texturé et chacune de ses projections sur les images I_3, I_4 et I_5 sont calculées. L'erreur de texturer le triangle t avec la texture I_1 i.e. la métrique de photocohérence relative à la

texture I_1 est la somme de toutes ces erreurs quadratiques. Ce procédé est répété pour les textures I_2, I_3, I_4 et I_5 et la métrique de photocohérence est donc calculée pour chacune de ces textures. La texture minimisant cette métrique est considérée comme meilleure texture pour le triangle t. Voir [14] pour plus de détails sur les algorithmes utilisés.

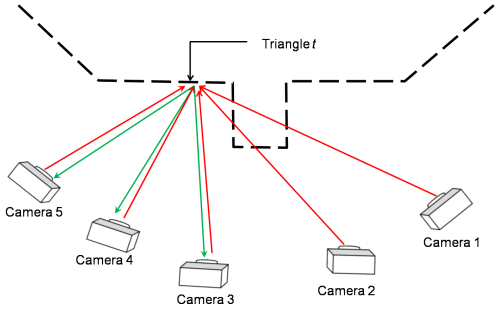


Figure 6 – *Plaquage de texture en utilisant la métrique de photocohérence. Par simplicité, le maillage à texturer est dessiné en pointillés représentant les triangles du maillage.*

4 Résultats

Les résultats sont présentés pour deux séquences. La séquence *Breakdancers*, fournie par Microsoft, a été utilisée par l'ancien groupe FTV de MPEG. Cette séquence est capturée par huit caméras disposées en arc. Les vues ne sont pas rectifiées. Les cartes de profondeur sont de bonne qualité. La deuxième séquence *Balloons* est fournie par l'Université de Nagoya et partagée dans le groupe MPEG 3DV. Les vidéos ont été capturées avec des caméras parallèles, et les vues sont rectifiées. Les cartes de profondeur sont de moins bonne qualité que celle de *Breakdancers*, néanmoins elles représentent un matériel plus réaliste pour une application pratique. Les deux séquences sont à la résolution XGA (1024x768).

4.1 Modélisation géométrique

La figure suivante illustre la surface obtenue après notre modélisation géométrique avec une résolution volumétrique moyenne.
On distingue des artefacts dans les zones de discontinuités de profondeur. A de telles résolutions, la quantification de profondeur présente aussi des artefacts notamment sur les surfaces planes. Ces deux types d'artefacts sont atténués grâce à notre algorithme de plaquage de texture.

4.2 Plaquage des textures

Calcul de distorsion : La distorsion a été calculée entre les images rendues et les images originales en utilisant le PSNR. Il est important de souligner que la distorsion est calculée uniquement sur les triangles visibles par au moins une caméra.
Même si la distorsion en termes d'erreur quadratique est grande, les résultats en termes de qualité visuelle sont plutôt satisfaisants. Grâce à notre algorithme de plaquage de texture, notre schéma de modélisation est robuste à la qualité des cartes de profondeur. Néanmoins, il est évident qu'une estimation plus exacte des cartes de profondeur donnerait de meilleurs résultats notamment dans la phase de modélisation géométrique.

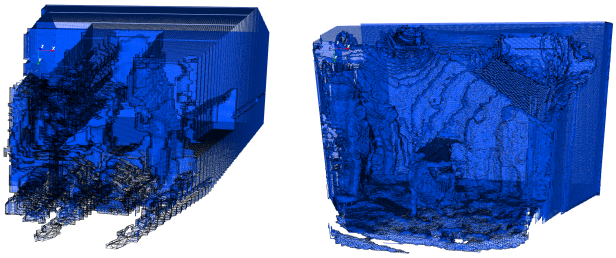


Figure 7 – *Résultats de notre schéma de fusion avec une résolution volumétrique de 250x250x250. A gauche : le modèle issu de Balloons. A droite : le modèle issu de Breakdancers.*

Figure 8 – *Résultats du plaquage de texture basé photocohérence pour Balloons. De haut en bas et de gauche à droite l'image de référence correspondant à la caméra 1, l'image de référence correspondant à la caméra 5, l'image synthétisée correspondant à la caméra 1 et l'image synthétisée correspondant à la caméra 5.*

5 Conclusion

Dans cet article, le problème de modélisation 3D de séquences vidéos à partir de données Mulit-vue plus Profondeur (MVD) a été abordé, sans aucune hypothèse sur la configuration des caméras. Un nouveau schéma de modélisation géométrique permettant la fusion des cartes de profondeur en un seul maillage triangulaire a été présenté. A cet égard, une représentation volumétrique reposant sur un algorithme de *Space Carving* itératif a été utilisé. Les cartes de profondeur ont été représentées par des voxels

Figure 9 – *Résultats du plaquage de texture basé photo-cohérence pour Breakdancers. De haut en bas et de gauche à droite : l'image de référence correspondant à la caméra 0, l'image de référence correspondant à la caméra 7, l'image synthétisée correspondant à la caméra 0, l'image synthétisée correspondant à la caméra 7.*

Sequence	PSNR (dB)		
Breakdancers	Camera 0	Camera 4	Camera 7
	29.52	33.51	30.27
Balloons	Camera 1	Camera 3	Camera 5
	28.94	31.83	30.92

Tableau 1 – *Distorsion des images synthétisées pour les séquence breakdancers et balloons.*

dont le statut (opaque ou transparent) est mis à jour selon leur cohérence géométrique avec ces cartes de profondeur. La frontière du modèle volumétrique est finalement convertie en maillage triangulaire grâce à l'algorithme de *Marching Cubes*. Un nouvel algorithme de plaquage de textures multi-vues a été ensuite proposé. Cet algorithme attribue à chaque triangle de la surface fusionnée la texture optimale en minimisant les distorsions entre les vues synthétisées et les vues originales. Le modèle 3D proposé et composé de texture et de géométrie nous permet de synthétiser toute vue se situant entre les caméras initiales avec peu d'artéfacts visibles. Comme perspective, le problème de la transmission de notre modèle 3D sera abordé en réduisant le coût de codage du modèle géométrique et en formalisant le signal de texture à transmettre.

Références

[1] C. Fehn, P. Kauff, M.O. De Beeck, F. Ernst, W. Ijsselsteijn, M. Pollefeys, L. Van Gool, E. Ofek, et I. Sexton. An evolutionary and optimised approach on 3d-tv. Dans *Proc. of IBC*, volume 2, pages 357–365, 2002.

[2] P. Merkle, A. Smolic, K. Muller, et T. Wiegand. Multi-view video plus depth representation and coding. Dans *ICIP 2007*, volume 1, pages I–201. IEEE, 2007.

[3] R. Balter, P. Gioia, et L. Morin. Scalable and efficient video coding using 3-d modeling. *Multimedia, IEEE Transactions on*, 8(6) :1147–1155, 2006.

[4] K. Mueller, A. Smolic, P. Merkle, B. Kaspar, P. Eisert, et T. Wiegand. 3d reconstruction of natural scenes with view-adaptive multi-texturing. Dans *3DPVT 2004*, pages 116–123. IEEE, 2004.

[5] M. Waschbüsch, S. Würmlin, D. Cotting, F. Sadlo, et M. Gross. Scalable 3d video of dynamic scenes. *The Visual Computer*, 21(8) :629–638, 2005.

[6] L. He, J. Shade, S. Gortler, et R. Szeliski. Layered depth images. Dans *Proceedings of the 25th annual conference on computer graphics and interactive techniques (SIGGRAPH 1998), July*, pages 19–24, 1998.

[7] M. Waschbüsch, S. Würmlin, et M. Gross. 3d video billboard clouds. Dans *Computer Graphics Forum*, volume 26, pages 561–569. Wiley Online Library, 2007.

[8] T. Colleu, S. Pateux, L. Morin, et C. Labit. A polygon soup representation for multiview coding. *Journal of Visual Communication and Image Representation*, 21(5-6) :561–576, 2010.

[9] W.N. Martin et JK Aggarwal. Volumetric descriptions of objects from multiple views. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (2) :150–158, 1983.

[10] K.N. Kutulakos et S.M. Seitz. A theory of shape by space carving. *International Journal of Computer Vision*, 38(3) :199–218, 2000.

[11] B. Curless et M. Levoy. A volumetric method for building complex models from range images. Dans *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312. ACM, 1996.

[12] S. Izadi et al. Kinectfusion : real-time 3d reconstruction and interaction using a moving depth camera. Dans *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568. ACM, 2011.

[13] W.E. Lorensen et H.E. Cline. Marching cubes : A high resolution 3d surface construction algorithm. *ACM Siggraph Computer Graphics*, 21(4) :163–169, 1987.

[14] Y. Alj, G. Boisson, P. Bordes, M. Pressigout, et L. Morin. Space carving mvd sequences for modeling natural 3d scenes. volume 8290, page 829005. SPIE, January 2012.

MULTI-TEXTURING 3D MODELS: HOW TO CHOOSE THE BEST TEXTURE?

Youssef Alj, Guillaume Boisson, Philippe Bordes, Muriel Pressigout, Luce Morin

MULTI-TEXTURING 3D MODELS: HOW TO CHOOSE THE BEST TEXTURE?

Youssef Alj, Guillaume Boisson, Philippe Bordes

Technicolor
1 avenue Belle Fontaine CS 17616
35576 Cesson-Sévigné Cedex
France

Muriel Pressigout, Luce Morin

INSA Rennes
20 avenue des Buttes de Coësmes
35043 Rennes Cedex
France

ABSTRACT

In this article, the impact of 2D based approaches for multi-texturing 3D models using real images is studied. While conventional 3D based approaches assign the best texture for each mesh triangle according to geometric criteria such as triangle orientation or triangle area, 2D based approaches tend to minimize the distortion between the rendered views and the original ones. Evaluation of the two strategies is performed on real scenes for two image sequences and results are provided using the PSNR metric. Moreover, an improvement of the image-based approach is proposed for texturing partially visible triangles.

Index Terms— Multi-texturing, best texture, OpenGL, mesh, PSNR.

1. INTRODUCTION

Since the introduction of texture mapping in computer graphics in the mid seventies, acquiring photo realistic 3D models has become necessary for various applications such as cultural heritage, architecture and entertainment industry. Typically in such applications, one seeks to texture a 3D model given a set of photographs captured from various viewpoints. The straightforward approach for texturing the 3D model is to determine the best texture for each mesh facet. The point is then to find a criterion to define the best texture. 3D geometric criteria such as angle between the line of sight and the triangle orientation [3], distance to the viewing camera [2] or facet's area [8] may be considered to solve this problem. The aim of multi-texturing is to determine the texture providing the best rendering. To this end, photoconsistency based multi-texturing was introduced in [1]. It aims to assign for each mesh triangle the best texture by minimizing the 2D distortion, *i.e*. the quadratic error between the rendered and original views. Given these approaches, we attempt to address the following questions : what is the impact of an approach based on 2D distortion on the quality of the rendered views ? And do 2D approaches really add a significant improvement in terms of PSNR ? In this paper we present a comparison between texture selection using 3D geometric criteria and a 2D approach based on distortion minimization of the rendered images. This distortion is measured using the PSNR metric classically used within the video coding community. Actually, the PSNR metric is employed in order to evaluate fidelity between a reference and recovered image. In this study, this metric is used for both assigning the best texture to mesh triangles as well as for comparing geometric versus image based criteria for multi-texturing 3D models. The contributions of this paper are twofold :

- First, we study the impact, on the rendered views, of using geometric versus image based criteria for multi-texturing 3D models.
- Second, we extend the photoconsistency based multi-texturing method to handle partially visible triangles.

2. RELATED WORK

View Dependent Texture Mapping was first introduced by Debevec et al. in [3] with real time application in [4]. In this scheme, the user specifies a virtual view to render the mesh, and the best texture for each mesh triangle is chosen as the one the minimizes the angle between the triangle normal and the viewing directions of input cameras. For instance, when generating a novel view from the left, it would be better to use the image corresponding to the left view as the texture map. View Dependent Texture Mapping approaches consider the transmission of the set of all textures which requires a tremendous amount of data. The Floating Textures scheme [5] alleviates inaccuracies inherent to the geometric model and the camera calibration, and proposes a pairwise warping between input images using optical flow. Instead of using the optical flow, Harmonised Texture Mapping [9] used an optimized version of the input mesh in order to control the range of texture deformation. Lempitsky and Ivanov [8] formulate multi texturing the 3D model as a labelling problem : each texture should be assigned as a label to the mesh triangle. They expressed their labelling strategy as on energy minimi-

zation problem with two terms, the first one describing how better an input texture suits the triangle and the second one penalizing seams' visibility. Computation of the best texture per triangle is based on the angle between the viewing direction and the triangle normal. Based on energy minimization scheme, Gal *et al*. [6] extend the best texture search space to include a set of transformed images that compensate geometric errors and make use of Poisson blending to address lighting variation. The Unstructured Lumigraph Rendering method [2] uses a penalty function for each view in order to determine the best view for each triangle, this penalty is a linear combination of three terms : visibility constraints, the angle between the desired view to render and the set of input cameras and the resolution of the projected triangle. Other approaches form a single texture map by blending the available images. In [10], the unique texture map generation is presented as a super resolution problem [7]. The authors used image processing tools to compute the correct weights for blending. Finally, a 2D criterion for best texture selection is based on photoconsistency with the set of input images was first proposed in [1]. Their idea is based on distortion minimization between the rendered and original views. This distortion is measured in terms of quadratic error.
In this paper, we compare the impact of multi-texturing 3D models using geometric criteria, namely triangle orientation and triangle area, versus using 2D criterion based on photoconsistency on the other hand.

3. GEOMETRIC CRITERIA FOR MESH TEXTURING

Given a triangular mesh $\mathcal{M}$ and a set of texture images $\mathcal{I} = \{I_1, \ldots, I_n\}$. Each texture I_i is respectively captured from the views $\mathcal{V} = \{V_1, \ldots, V_n\}$. The aim is to find for each mesh triangle $t \in \mathcal{M}$ the best texture image $\hat{\mathrm{I}}_t \in \mathcal{I}$ using a geometric criterion C_k, where C_k could be one of the following criteria :

- $C_1 := \langle \vec{d_j}, \vec{n}_t \rangle$ *i.e*. dot product between triangle normal $\vec{n}_t$ and viewing direction (*i.e*. optical axis, that is image plane normal) $\vec{d_j}$ of the view V_j. Small angles between triangle normal and viewing directions of input cameras are then preferred. Note that in the case of parallel camera setup this criterion is not applicable.
- $C_2 := \langle \vec{e}_j, \vec{n}_t \rangle$, where $\vec{e}_j$ denotes the direction of the ray joining the triangle barycenter and the camera's position. Views where the triangle's viewing cone is larger are then preferred.
- $C_3 := \mathrm{area}_j(t)$ the area of the projected triangle onto the view V_j. Views where the resolution of the projected triangle is larger are then preferred.

Let $P_j(t)$ denote the projection of triangle t onto the view V_j. $P_j(t)$ consists in a set of pixels (q, l) corresponding to the rasterization of triangle t, *e.g*. using *OpenGL* rendering engine. The best texture $\hat{\mathrm{I}}_t$ is computed the following way : first for each triangle t, total and partial visibility are determined with respect to each view. The best texture is then determined as the maximizer of C_k with respect to visibility constraints. Figure 1 illustrates the principle of best texture selection using C_1. This figure shows that the best view to texture the triangle t is the camera 2. Indeed, the angle between the viewing direction of the camera 2 and the triangle normal is minimum, which corresponds to a maximum dot product.

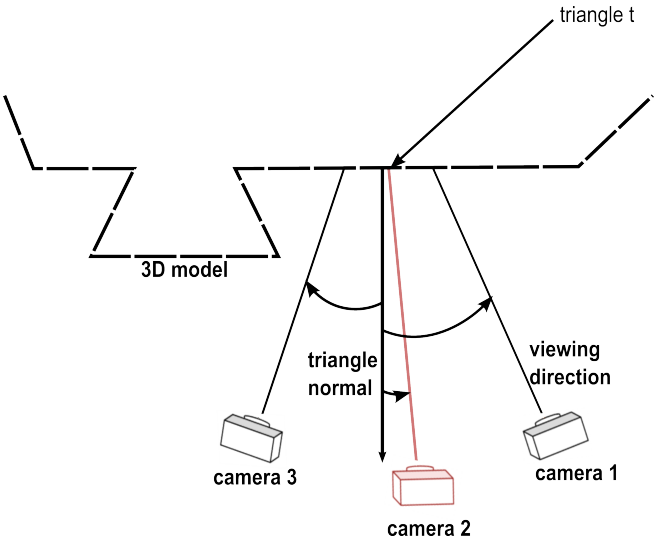


Fig. 1. Best texture selection as the minimizer of the angle between the triangle normal and cameras viewding direcions.

3.1. Visibility determination

Given the input mesh $\mathcal{M}$, total and partial visibility are first determined for each triangle with respect to each view. At the end of this step, each triangle will have one of the three labels per view : "totally visible", "partially visible" or "hidden". The algorithm is detailed in the next section.

3.1.1. Total visibility

In this step only the labels "totally visible" and "hidden" are assigned to mesh triangles. The status of each mesh triangle is initialized to "totally visible". The visibility of each triangle is determined by computing the visibility of the triangle vertices. A triangle is marked as hidden if at least one of its vertices is hidden. Vertex visibility with respect to each camera is determined using OpenGL z-buffer denoted as Z^j_{buffer} (*i.e*. z-buffer of the mesh projected onto the view V_j). During the first pass, the input mesh is projected onto the current view, and the z-buffer is extracted. The second pass is dedicated to vertex visibility determination using the computed z-buffer. Each mesh vertex is projected onto the current view and the depth component, denoted as $z_{\mathrm{projected}}$, is checked

against the pixel depth $Z^j_{\text{buffer}}[q,l]$. If the projected vertex is behind the pixel in the z-buffer, then this vertex is hidden, and thus the set of mesh triangles sharing this vertex are marked hidden. The pseudo-code for total visibility determination is provided in algorithm 1.

Algorithm 1: Global visibility determination.

```
//  Views traversal
for each view V_j do
    Initialize all triangles to visible.
    Project the M onto V_j.
    Store the depth buffer Z^j_buffer.
    // Mesh vertices traversal
    for each vertex v do
        Determine (q, l, z_projected) the projection of the
        vertex v onto V_j.
        if Z^j_buffer[q, l] > z_projected then
            v is a hidden vertex.
            Mark all the triangles lying on v as hidden.
```

3.1.2. Partial visibility

Until now, each triangle has one of the two labels : "totally visible" or "hidden". At the end of this step, "hidden" triangles will be sorted out as "partially visible" or totally "hidden". To this end the depth of each pixel is checked against the stored depth buffer in the projected triangle. Note that it is necessary to check the depth of the projected triangle. Partial visibility determination by counting the number of hidden vertices is not sufficient as some triangles could be partially visible and have a number of hidden vertices of three. The algorithm determining partial visibility is presented in algorithm 2.

Algorithm 2: Partial visibility determination.

```
//  Views traversal
for each view V_j do
    // Mesh triangles traversal
    for each triangle t do
        if triangle t is hidden in V_j then
            Determine P_j(t) the projection of triangle t
            onto V_j.
            for each pixel (q, l) in P_j(t) do
                Get the pixel depth z.
                if z == Z^j_buffer[q, l] then
                    Mark t as partially visible in V_j.
```

3.2. Best texture computation using geometric criterion

In this section the best texture is assigned to the set of triangles which are totally or partially visible using one of the three criteria discussed above. Algorithm 3 describes how the best texture is assigned for such triangles. The user specifies a geometric criterion C_k. The view that maximizes C_k is assigned as the best texture for the current triangle.

Algorithm 3: Best texture determination using C_k.

```
//  Triangles traversal
for each triangle t do
    Get the normal of the triangle t.
    //  Views traversal
    for each view V_j do
        if t is visible or partially visible in V_j then
            Compute C_k.
    Compute Î_t which maximizes C_k
```

4. PHOTO-CONSISTENCY BASED MESH TEXTURING

Photoconsistency based mesh texturing was first introduced in [1]. Figure 2 illustrates the photoconsistency principle. The triangle t is visible in cameras 1, 2, 3. For each texture $I_i \in \mathcal{I}$, the triangle t is textured with texture I_i (red arrow for texture mapping operation) and projected (green arrow for projection operation) onto the cameras 1, 2, and 3 - i.e. the set of cameras where t is visible. The error of texturing the triangle t with each available texture is computed. This error is referred to as the photoconsistency metric. The best texture is chosen as the minimizer of the photoconsistency metric. In [1] only totally visible triangles are addressed, and partially visible ones are textured using the view of the frontal camera. The novelty of the approach described in this paper is to extend the photoconsistency-based best texture computation for partially visible triangles too. The overall framework is sketched in Figure 3. First, visibility is determined for each triangle with respect to each view. Second, each available texture is mapped on visible triangles of the input mesh. Photometric projection error is computed for each triangle and for each input view and stored in distortion images. Synthesized views are produced by rendering the textured mesh, each triangle being textured with the best texture according to the photoconsistency metric. In the next section each step of this algorithm is described in more details.

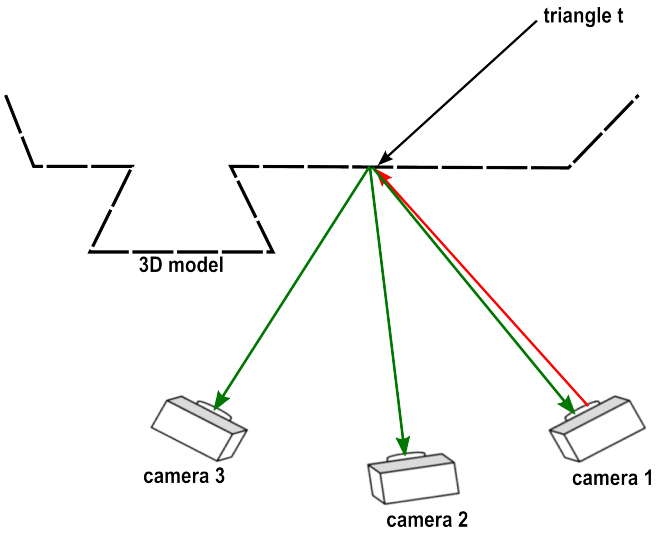


Fig. 2. Best texture determination using photoconsistency. Red arrow for texture mapping operation and green arrow for projection operation.

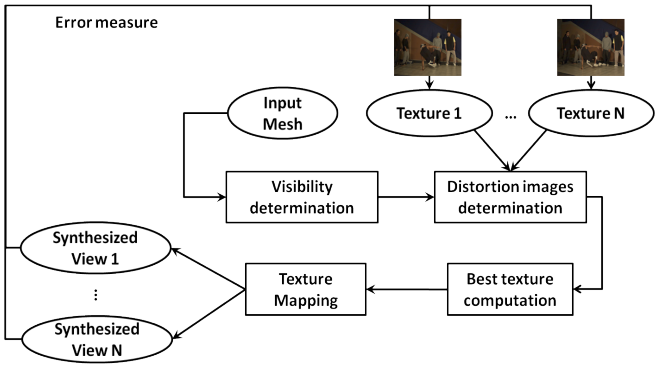


Fig. 3. Overall framework.

4.1. Distortion images determination

In order to determine the error of texturing a triangle with an input texture I_i, each available texture is then mapped on the merged mesh and projected onto each camera. Let $P_j^i(\mathcal{M})$ be the projection of the mesh $\mathcal{M}$ onto the view V_j textured with the image I_i. For each available texture the following distortion image is computed in RGB color space :

$$D_{i,j} = \|P_j^i(\mathcal{M}) - I_j\|_2$$

4.2. Best texture determination

In this section, the best texture is chosen for each mesh triangle using the distortion images. The best texture for each triangle relies on the computed set of pixels that belong to the projected triangle and visible in the current view V_j, i.e. the set of pixels :

$$\text{Vis}_j(t) = \{(q,l) \in P_j(t), (q,l) \text{ visible in } V_j\}$$

Figure 4 shows how $\text{Vis}_j(t)$ is determined for partially and totally visible triangles. The grid represents image pixels. In 4(a) the blue triangle is totally visible, the set of pixels defining $\text{Vis}_j(t)$ are shown in red. On the contrary, the blue triangle in 4(b) is partially visible as it is occluded by the green triangle. $\text{Vis}_j(t)$ in this case is restricted to the non-occluded pixels. The photoconsistency metric is then computed for each triangle with respect to $\text{Vis}_j(t)$. This metric measures the error of texturing the triangle t with the texture image I_i. From here on, for sake of simplicity, visible triangle will refer to totally or partially visible triangle as they will be treated the same way.

$\forall i \in \{1, \ldots, n\}$ we compute :

$$\mathcal{E}_{t,I_i} = \sum_{\substack{j=1 \\ t \text{ visible in } V_j}}^{n} \left(\sum_{(q,l) \in \text{Vis}_j(t)} D_{i,j}[q,l] \right) ,$$

The best texture for a visible triangle t is then given by :

$$\hat{I}_t = \arg\min_{I_i \in \mathcal{I}} \mathcal{E}_{t,I_i} .$$

The steps of best texture computation are summarized in algorithm 5.

Algorithm 4: Compute distortion images.

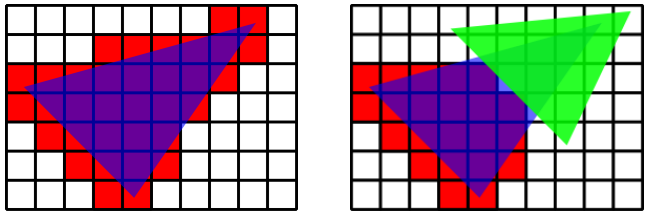
```
// Textures traversal
for each texture I_i do
    // Views traversal
    for each view V_j do
        Compute P_j^i(M) : projection of the mesh
        textured with texture I_i onto view V_j.
        D_{i,j} = ||P_j^i(M) − I_j||_2
```

Algorithm 5: Compute the best texture for each triangle.

```
// Triangles traversal
for each triangle t do
    // Views traversal
    for each view V_j do
        Determine Vis_j(t).
    // Textures traversal
    for each texture I_i do
        Compute E_{t,I_i}.
    Determine Î_t.
```

4.3. Texture mapping

Finally, the views are rendered. The 3D model is projected according to the camera's parameters supplied by the user. Then each triangle is textured using the texture coordinates determined during visibility determination step.



(a) total visibility (b) partial visibility

Fig. 4. Best texture computation. Handling partial/global visibility.

5. IMPLEMENTATION AND RESULTS

Results are presented for two sequences *breakdancers* and *balloons*. Breakdancers, provided by Microsoft, is captured by eight convergent cameras. Balloons sequence, provided by Nagoya University, is captured by three parallel cameras. Both sequences have XGA resolution (1024x768). The geometric model used in this study is based on the 3D reconstruction scheme presented in [1].

Results of the rendered views are provided in figure 6 and in figure 5. It can be seen that photoconsistency based mesh texturing provides better visual quality than geometric based methods. More precisely, in figure 5, the frontier between the two brown walls is well-aligned using the photoconsistency criterion, while this frontier is misaligned using the geometric criterion. Besides, the floor appears to be noisy using geometric criterion, while it is much smoother using the photoconsistency criterion. Figure 7 shows that the difference in terms of PSNR between photoconsistency criterion and any other geometric criterion is noticeable (the average difference is at least 1dB between the two methods).

Regarding the photoconsistency criterion, results also show that PSNR is higher for cameras near the frontal camera (see Figure 7), which is due to the large number of triangles assigned to the texture of the frontal camera. However, photoconsistency based multi-texturing algorithm is time-consuming (about 10x more) comparing to geometric methods. Therefore it turns out that photoconsistency based multi-texturing is best suited for offline rendering. These results suggest the use of the photoconsistency based multi-texturing for transmission purposes where distortion minimization is often a key element to take into account.

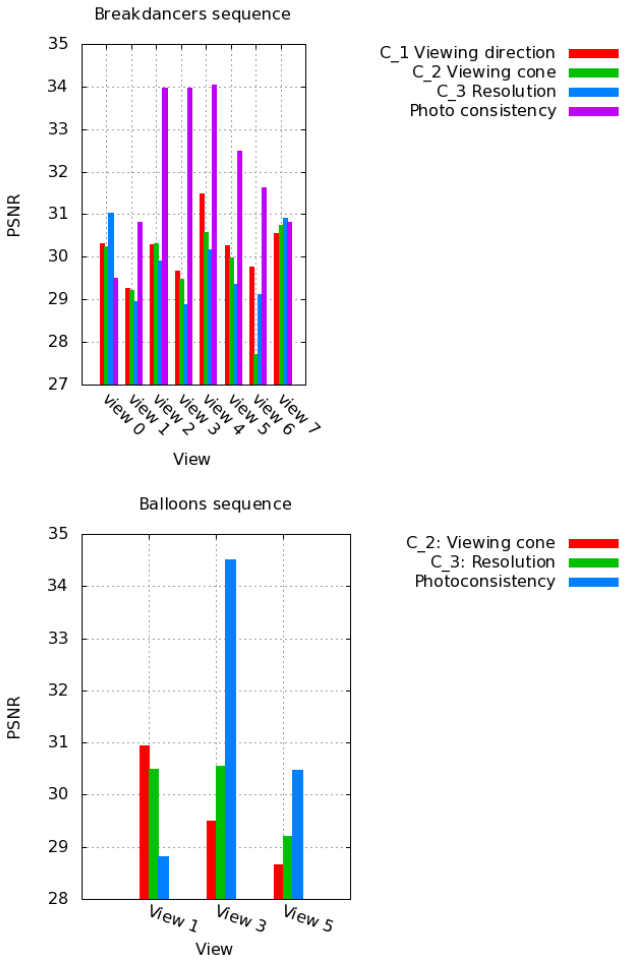


Fig. 7. PSNR evolution through different views using different criteria for Breakdancers and Balloons sequences.

6. CONCLUSION

In this paper, multi-texturing 3D models was addressed by considering several criteria for best texture assignment to mesh triangle. Conventional geometric criteria assign the view that minimizes the angle between the viewing direction and the triangle normal, or the view that maximizes the triangle resolution. On the contrary, 2D based approaches minimize the distortion between the rendered views and the original ones. We show that photoconsistency based methods are significantly more relevant for transmission purposes. Indeed they outperform geometric based methods both in terms of objective and subjective visual quality of synthesized views. The PNSR gap exceeds 2dB for the common test-sequences. As a perspective, we plan to investigate how to reduce complexity without altering too much the quality of the synthesized views. We also aim to improve spatial consistency of the assigned textures in order to reduce the coding cost of the

texture signal.

7. REFERENCES

[1] Y. Alj, G. Boisson, P. Bordes, M. Pressigout, and L. Morin. Space carving mvd sequences for modeling natural 3d scenes. volume 8290, page 829005. SPIE, January 2012.

[2] C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen. Unstructured lumigraph rendering. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 425–432. ACM, 2001.

[3] P.E. Debevec, C.J. Taylor, and J. Malik. Modeling and rendering architecture from photographs : A hybrid geometry-and image-based approach. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 11–20. ACM, 1996.

[4] P.E. Debevec, Y. Yu, and G. Borshukov. Efficient view-dependent image-based rendering with projective texture-mapping. In *Eurographics Rendering Workshop*, volume 98, pages 105–116, 1998.

[5] Martin et al. Eisemann. Floating textures. *Computer Graphics Forum (Proc. of Eurographics)*, 27(2) :409–418, April 2008.

[6] R. Gal, Y. Wexler, E. Ofek, H. Hoppe, and D. Cohen-Or. Seamless montage for texturing models. In *Computer Graphics Forum*, volume 29, pages 479–486. Wiley Online Library, 2010.

[7] M. Iiyama, K. Kakusho, and M. Minoh. Super-resolution texture mapping from multiple view images. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 1820–1823. Ieee, 2010.

[8] V. Lempitsky and D. Ivanov. Seamless mosaicing of image-based texture maps. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–6. IEEE, 2007.

[9] T. Takai, A. Hilton, and T. Matsuyama. Harmonised texture mapping. In *Proc. of 3DPVT*, 2010.

[10] L. Wang, S.B. Kang, R. Szeliski, and H.Y. Shum. Optimal texture map reconstruction from multiple views. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–347. IEEE, 2001.

(a) Original image camera 7

(b) Zoom on original image camera 7

(c) Synthesized image camera 7 using viewing cone criterion C_2

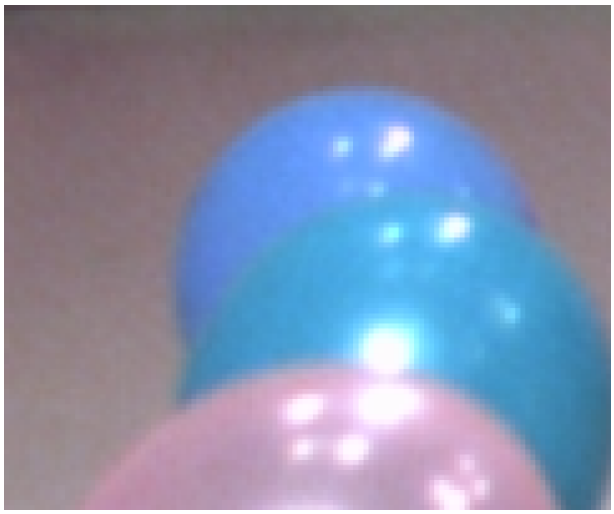
(d) Zoom on synthesized image camera 7 using viewing cone criterion C_2

(e) Synthesized image camera 7 using photoconsistency

(f) Zoom on synthesized image camera 7 using using photoconsistency

Fig. 5. Rendered images for breakdancers sequence using viewing cone criterion vs. photoconsistency.

(a) Original image camera 5



(b) Zoom on original image camera 5

(c) Synthesized image camera 5 using triangle resolution criterion C_3

(d) Zoom on synthesized image camera 5 using triangle resolution criterion C_3

(e) Synthesized image camera 5 using photoconsistency

(f) Zoom on synthesized image camera 5 using photoconsistency

Fig. 6. Rendered images for balloons sequence using resolution criteria versus photoconsistency.

Space Carving MVD Sequences for Modeling Natural 3D Scenes

Youssef Alj, Guillaume Boisson, Philippe Bordes, Muriel Pressigout, Luce Morin

Space Carving MVD Sequences for Modeling Natural 3D Scenes

Youssef Alj[a,b], Guillaume Boisson[a], Philippe Bordes[a], Muriel Pressigout[b], Luce Morin[b]

[a]Technicolor, Cesson-Sévigné, France
[b]Institut d'Electronique et des Télécommunications de Rennes (IETR), Rennes, France

ABSTRACT

This paper presents a 3D modeling system designed for Multi-view Video plus Depth (MVD) sequences. The aim is to remove redundancy in both texture and depth information present in the MVD data. To this end, a volumetric framework is employed in order to merge the input depth maps. Hereby a variant of the Space Carving algorithm is proposed. Voxels are iteratively carved by ray-casting from each view, until the 3D model be geometrically consistent with every input depth map. A surface mesh is then extracted from this volumetric representation thanks to the Marching Cubes algorithm. Subsequently, to address the issue of texture modeling, a new algorithm for multi-texturing the resulting surface is presented. This algorithm selects from the set of input images the best texture candidate to map a given mesh triangle. The best texture is chosen according to a photoconsistency metric. Tests and results are provided using still images from usual MVD test-sequences.

Keywords: Multi-view Video plus Depth (MVD), 3DTV, FTV, depth map fusion, Space Carving, multi-view texture mapping.

1. INTRODUCTION

Multi-View Imaging (MVI) has gained an unceasingly growing interest in the last few years. Used with stereoscopic displays, MVI provides a realistic depth perception to the user and allows a virtual navigation around the scene. It also offers the possibility to synthesize virtual views, opening therefore a broad spectrum of research topics and applications such as 3DTV and Free-viewpoint TV (FTV). To this end, depth estimation from stereoscopic matching has been widely investigated. Nevertheless, several technical deficiencies hampered a successful introduction of these applications to the mass market. Actually an end-to-end MVI chain presents many challenges due to the inherent problems related to scene capture, data representation and transmission. Indeed, during scene acquisition, multiple cameras are used, hence several photometric errors may occur due to the changing illumination across different views, which increases the signal ambiguity during depth estimation. Scene representation is also a challenging task. One should strive to build a dense yet non-redundant scene model from the set of overlapping views. The compromise between model simplicity and easy rendering is usually hard to satisfy. The representation is then to be transmitted through a communication channel. Determining the proper rate-distortion trade-off is another key point. At decoder side, high fidelity to original views is required. Finally to assess the quality of virtual views, conventional image-based objective metrics such as Peak of Signal to Noise Ratio (PSNR) or Structural Similarity (SSIM) are not applicable since the reference images for such views are not available.

In the context of scene representation, research activities have been strengthened worldwide in order to overcome the issues discussed above. Scene representation methods can be classified into three categories : image-based representations, geometry-based representations and intermediate representations.

Image-based representations : Earlier attempts tried to extend image processing tools to 3D scene representation. A scene is simply represented using the set of captured images. This is the case in Multi View Video (MVV) representation. The 2D+Z[1] scheme uses a single view in addition to a depth image, also known as depth map, representing the depth information of each object of the scene. In this paper we consider Multi-view Video plus Depth representation (MVD)[2] as input, each view being composed of the captured image and the corresponding depth map.

Geometry-based representations : In the computer vision community, one of the core tasks is to extract 3D information from images. Thus many algorithms have been developed to extract a 3D model from multi-view data. On the other hand, the computer graphics community is developing IBR (Image Based Rendering) that is modeling of 3D scenes partly using real images as (one of the) input data. A 3D model extracted from the images may also be used for multi-view representation and compression. The extracted 3D model may be mesh-based,[3] voxel-based,[4] or point-based.[5]

Intermediate representations : Layered Depth Images[6] (LDI) were introduced in order to reduce redundancy in MVD material, by selecting a reference view and adding occluded information contained in adjacent views to the representation. 3D video billboards approach[7] extends the work developed in[8] and uses a hybrid approach combining depth maps and textured mapped rectangles intended to represent detailed geometry of the scene. Polygon soup representation was developed in[9] and uses a set of disconnecting and overlapping 3D polygons as a compact representation of the scene.

Scene representation has become an active research area with a wide variety of applications. Hence the choice of scene representation depends on the aimed application. For 3D video applications[10] , image-based representation are more popular since high fidelity with respect to original data is often required at the rendering stage. For mixed/augmented reality applications, geometric-based modeling methods are the most ubiquitous approaches. Typically, the purpose in such applications[11] is to recover an object of interest from a set of captured images. Foreground objects are extracted from input images, and it's not necessary to model the whole scene. High fidelity with respect to original views is therefore not necessarily satisfied, but they can achieve real time rendering with interactive user-experience.

In this paper, we take advantage of image-based representations, thanks to the use of MVD data, to ensure high fidelity to original data, and geometric-based representation to enforce the compactness of our scene representation.

The contributions of this paper are twofold. First, a new volumetric framework is proposed in order to merge the input depth maps into a single and compact surface mesh. This is the subject of section 3. Second, a new multi-view texture mapping using photoconsistency is introduced to texture the resulting surface mesh. This is presented in section 4. Last results are presented in section 5. But first in section 2 we will present a short state-of-the-art.

2. RELATED WORK

In this section, a short review of volumetric methods for scene representation is presented (see[12] and[13] for more detailed surveys).

Scene representation from a set of images has been an active research area for several decades. The aim is to build a 3D model given 2D views of the scene. Volumetric methods have known a successful introduction in computer vision community. Indeed they provide a robust and compact representation of the scene. Typically, these methods assume the existence of a bounding volume on which the scene of interest lies. This bounding volume is subdivided into cubical elements called *voxels*. During the process of *Space Carving* each voxel is labelled as *transparent* or *opaque*, depending on a consistency measure extracted from the input images. The volumetric representation of the scene is the set of opaque voxels. A surface mesh is then extracted from this volume using *Marching Cubes* algorithm. The consistency measure is based either on the silhouette information extracted from input images, which is referred to as *shape-from-silhouette* method, or according to the photometric information contained in the input images, which is referred to as *shape-from-photoconsistency* method. Alternative approaches use range images as input data and fuse them into a single surface.

Shape-from-silhouette : These methods require that foreground objects in the input images can be extracted from the background. Each object of interest is reconstructed from the set of silhouette images. Therefore the scene can be recovered by integrating each reconstructed objects with the background images. A silhouette image is a binary image where each pixel's value indicates whether the ray from the camera center to the input image

intersects an object of interest or not. The set of rays defines a silhouette cone. The intersection of the silhouette cones associated to all cameras is called the *visual hull*[14] . The visual hull of an object can also be defined as the maximal shape that gives the same silhouette as the actual object's silhouette for all views. A voxel is determined to be in the visual hull by projecting it into the set of all images and testing if it is inside the silhouette. To optimize scene space traversal, Potmesil[15] extends the regular-grid-based visual hull to octrees. Rather than using a voxel description, the authors in[11] use Delaunay tetrahedrization as a discretization of the scene space. The final surface is obtained by removing tetrahedrons lying outside the visual hull using the silhouette information. The major drawback of the shape-from-silhouette approach is that not all concavities present in the silhouette images can be represented.

Shape-from-photoconsistency : In such schemes, voxel consistency is measured directly using the captured images. This is the idea behind the Space Carving algorithm[16] . More precisely, a voxel is said to be photo-consistent with the set of input images if its projections onto the cameras in which this voxel is visible are all the same. In practice, a user-defined threshold is used in order to control the correlation between the projected voxels and the input data. This limitation has been overcome within the probabilistic framework developed by[17] . However, Space Carving suffers from another severe limitation. Indeed the surface to recover is assumed to be Lambertian. However natural scenes may contain areas with changing illumination effects due to image capturing conditions. Therefore, this assumption is usually not satisfied and this may significantly damage the relevance of the output volume by carving away wrong voxels. An approach combining the visual hull and photoconsistency constraints was developped by[18] . They formulated the problem of finding an optimal photoconsistent surface as an energy minimization problem. This cost function is discretized on a graph and the solution is obtained as the minimum cut solution of this graph.

Volumetric fusion of range images : Curless and Levoy proposed[19] a method for merging range images that relies on computing a weighted signed distance function cumulated respectively for each range image. The signed distance function takes 0-value for voxels containing points from acquired data. The distance function is computed from each voxel to the sensor and its value is stored for each voxel. For a given voxel, the so-called distance function is cumulated over each range image. Zero-crossing voxels correspond to the merged surface.

The proposed approach in this article is closer to range images fusion approaches since depth maps can be considered as range images. However as in the photo-consistency approach, the aim is to build a 3D model photoconsistent with the set of input images. The photoconsistency constraint is thus enforced using the proposed texture-mapping method.

3. GEOMETRY MODELING : A VOLUMETRIC FRAMEWORK FOR DEPTH MAPS FUSION

In this section, our volumetric framework intended to merge the input depth maps is presented. Considering a generic framework, the input sequences are calibrated but not necessarily registered. Epipolar lines may not be parallel to pictures lines. Consequently 3D reprojection operations may be more complex than horizontal shifts depending on objects' depth. The issue of the inter-view disparity estimation is not addressed in this paper, and the provided depth maps are assumed to be reliable. The purpose is to merge the input depth maps into a single surface mesh consistent with the original views.

An overview of our merging scheme is sketched in figure 1. First, for each provided depth map, a high resolution mesh is built. Second, the bounding volume enclosing the set of the built meshes is defined then discretized into voxels. A voxel may be either opaque or transparent. For each available camera, the set of voxels lying in its viewing cone are determined and labelled as opaque. The next step is dedicated to the Volumetric Mesh Hull (VMH) determination. Given a mesh corresponding to a viewpoint, the VMH defines the set of voxels that lie on this surface mesh. Our formulation of the Space Carving algorithm relies on the computed VMH. Indeed, for each available view, rays are cast from the camera center to each voxel of the corresponding VMH. The status of each voxel lying on this ray is then set to transparent and dedicated to be carved. Finally, a surface mesh is extracted from this binary-labelled volume using the Marching Cubes algorithm.

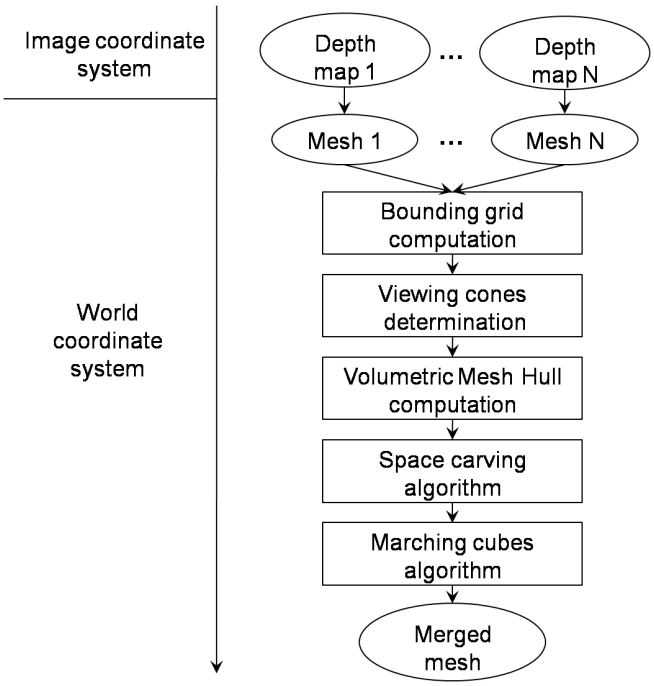


FIGURE 1. Our volumetric framework for depth maps fusion.

3.1 From depth maps to meshes

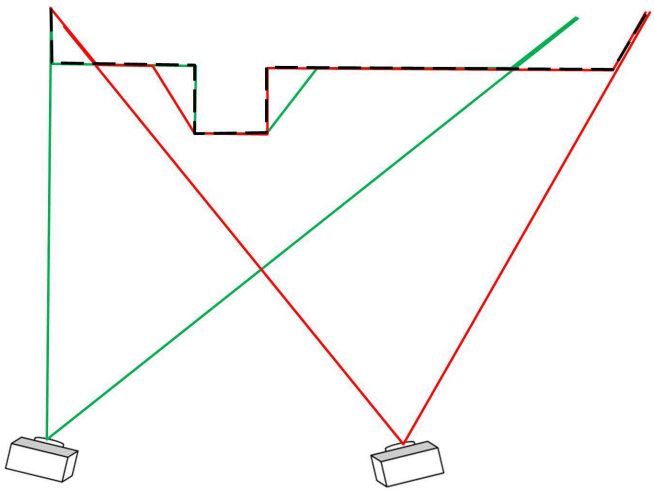


FIGURE 2. Example of two surfaces to be merged (one in green and the other in red). The expected surface as output of the algorithm is in black dotted line. Cameras viewing each mesh are also represented with their viewing cones.

The first step of our algorithm is the generation of one mesh for each of the provided depth maps. From each of the available views, a high resolution uniform mesh is built in the image domain. Each mesh vertex represents a pixel in the depth image. Registration is performed by converting each triangular mesh back to world coordinate system using equation (1).

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{pmatrix} & R & & T \\ & & & \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} f_x & S_{uv} & c_u & 0 \\ 0 & f_y & c_v & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}^{-1} \begin{pmatrix} su \\ sv \\ s \\ 1 \end{pmatrix} \tag{1}$$

Where, X, Y and Z are the coordinates of a 3D point in the world reference Coordinate System. u and v are the coordinates of the projected 3D point in image coordinate system, and $s = z_{cam}$ is the depth of the pixel (u, v) with respect to the camera Coordinate System.
R and T respectively denote the rotation matrix and the translation vector of the considered camera's Coordinate System in the world Coordinate System. f_x, f_y, S_{uv}, c_u and c_v are the camera's parameters.

3.2 Bounding grid computation

The set of meshes are now available for our volumetric representation. The basic data structure on which our volumetric framework operates is a volumetric grid. This grid is built by a regular subdivision of the bounding box enclosing the set of meshes into parallelepipedic elements called voxels. A binary labelling strategy is used to label the voxels. Each voxel can be either opaque or transparent. The label of each voxel grid is initialized to transparent. Voxels labels are modified throughout the algorithm according to their relevance to represent the scene.

3.3 Viewing cones determination

In this step, the purpose is to find the set of voxels lying in each camera's viewing cone. Voxels outside this viewing cone are labelled as transparent. The viewing cone of each camera is modeled by a frustum defined by four planes bounding the range of visible points according to each axis (see Figure 3). The task is then to compute for each camera the set of voxels that lie in its corresponding frustum. To this end, frustum planes equations are determined for each camera, these equations allow to perform the inside/outside test for each voxel center against the considered frustum.

Frustum planes equations : Equation (1) represents the conversion from image to world coordinates system. This conversion is invertible, and image coordinates can be expressed using the following equation :

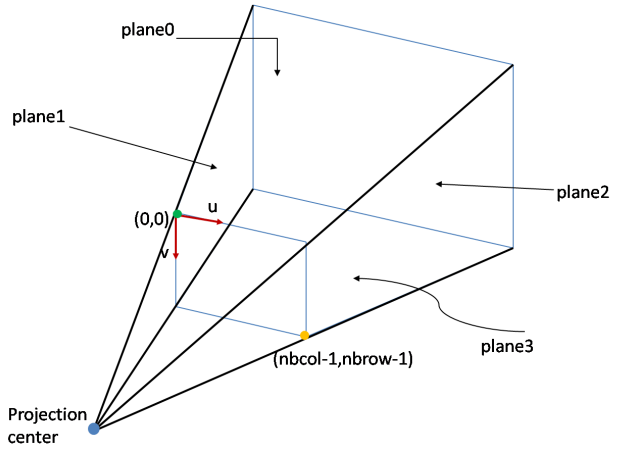


FIGURE 3. Frustum planes computation.

$$\begin{pmatrix} su \\ sv \\ s \end{pmatrix} = \begin{pmatrix} f_x & S_{uv} & c_u & 0 \\ 0 & f_y & c_v & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} & R^{-1} & & -R^{-1}.T \\ & & & \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix},$$

This equation can be reformulated by computing the product of the two central matrices :

$$\begin{pmatrix} su \\ sv \\ s \end{pmatrix} = \begin{pmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix},$$

The last equation leads to the two following parametric plane equations :

$$X(p_{11} - up_{31}) + Y(p_{12} - up_{32}) + Z(p_{13} - up_{33}) + p_{14} - up_{34} = 0 \quad (2)$$

$$X(p_{21} - vp_{31}) + Y(p_{22} - vp_{32}) + Z(p_{23} - vp_{33}) + p_{24} - vp_{34} = 0 \quad (3)$$

Equation (2) with $u = 0$ (resp. $u = \text{nbcol} - 1$) is the equation $E_1(X, Y, Z)$ of the vertical left-hand plane 1 (resp. the equation $E_2(X, Y, Z)$ of the right-hand plane 2) see figure 3.

Equation (3) with $v = 0$ (resp. $v = \text{nbrow} - 1$) determines the equation $E_3(X, Y, Z)$ of the upper plane 0 (resp. the equation $E_4(X, Y, Z)$ of the bottom plane 3).

Once planes equations are computed, inside/outside points with respect to each camera can be determined as the following : Given a point in the frustum $I(X_I, Y_I, Z_I)$

Inside condition test : $\forall i,\ 1 \leq i \leq 4$ compute $\text{sign}(E_i(X_I, Y_I, Z_I))$ where,

$$\text{sign}(x) = \begin{cases} +1 & \text{if } x \geq 0 \\ -1 & \text{otherwise} \end{cases}$$

Given a voxel defined by its center $V(X_V, Y_V, Z_V)$, if $\forall i,\ 1 \leq i \leq 4$,
$\text{sign}(E_i(X_I, Y_I, Z_I)) = \text{sign}(E_i(X_V, Y_V, Z_V))$ then this voxel is inside the frustum, otherwise this voxel is outside.

At the end of this step, all voxels belonging to the union of cameras viewing cones are labelled as opaque. All the remaining voxels within the bounding grid are still labelled transparent.

3.4 Building the Volumetric Mesh Hull (VMH)

This section describes the computation of the Volumetric Mesh Hull (VMH) which will be used during the space carving process. For each input mesh, the set of voxels intersecting the surface mesh defines the VMH associated with this mesh.

As an output each voxel has an additional information defining which VMH(s) this voxel belongs to.

For each mesh, each triangle is visited and scanned with respect to the voxel grid, i.e. this scan defines the voxels lying on this triangle. All these voxels are then marked to belong to the current VMH by updating the additional information associated with the voxel. In order to scan the 3D triangle surface, the classical 2D scan line algorithm is extended to the 3D case.

The 3D scan line algorithm is presented in algorithm 1.

```
//  Triangles traversal
for each triangle T do
    for each axis X, Y and Z do
        Compute the integer bounds b_min and b_max of T along the axis considered.
        for m ← b_min to b_max do
            Compute the intersection line of triangle T with the plane at coordinate m.
            Find voxels along this intersection line using 3D Bresenham line algorithm.
            Add these voxels to VMH.
        end
    end
end
```

Algorithm 1: Volumetric Mesh Hull (VMH) determination.

3.5 Space carving

Until now, an opaque voxel is a voxel lying in at least one camera's viewing cone. A VMH voxel is a voxel lying on one of the surface mesh. In this step, space carving is performed by updating voxels labels from opaque to transparent for all voxels "in front of each VMH". More precisely, each VMH is considered and a geometric consistency criterion is employed in order to update voxels labels. This criterion is based on the relative position of the voxel according to the considered surface mesh and the corresponding viewpoint. A voxel is said to be geometrically consistent with a surface mesh if it lies behind it with respect to the viewing camera. Voxels detected as geometrically inconsistent are labelled as transparent. The carving process is performed by iteratively updating the carved volume with respect to each camera. The volume to be carved is initialized to the union of voxels inside the set of viewing cones computed in section 3.3 (see figure 4). This volume is then refined iteratively with geometrical information lying in each view (see figures 4 and 5), carving occluding areas relative to each camera. For each of the available views, rays are cast from the viewpoint to each voxel of the camera's corresponding VMH. For each ray, voxels lying on the current ray are scanned using 3D Bresenham line retracing algorithm. Since they are geometrically inconsistent, these voxels are dedicated to be carved (i.e. updated to transparent).

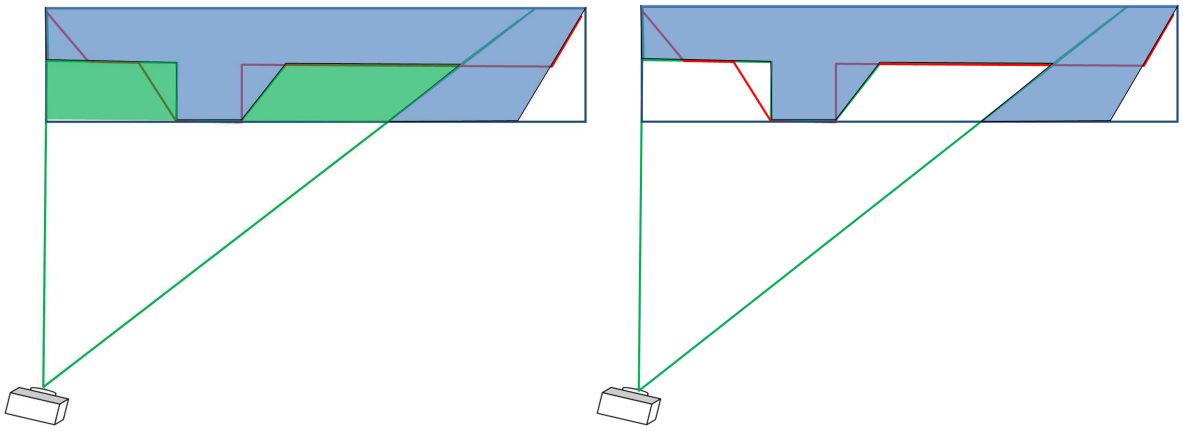


FIGURE 4. Space carving according to the left camera. Rays are cast from the camera center to the corresponding VMH (left). Green areas are carved. The result of this carving operation according to the green camera is shown on the right.

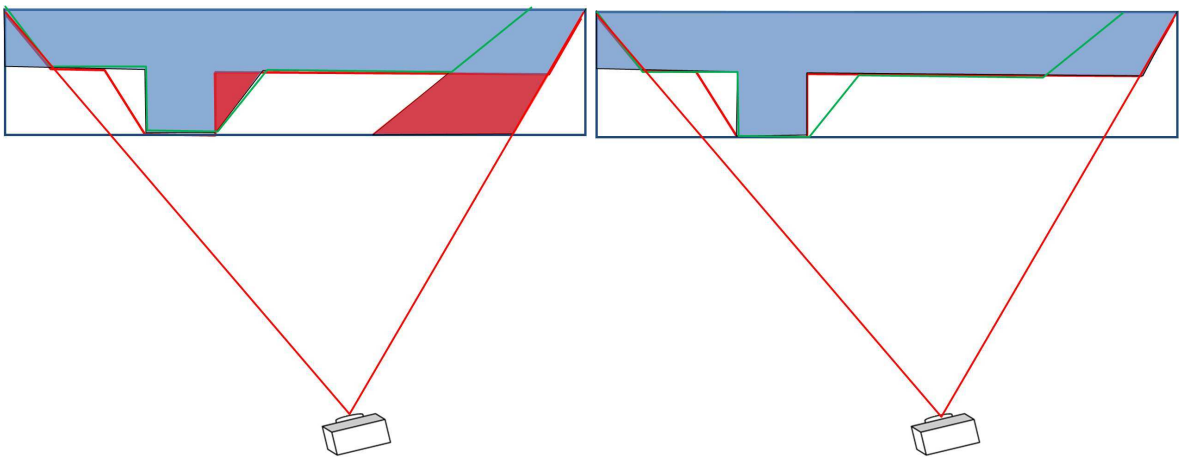


FIGURE 5. Space carving according to the right camera. For the carved volume to be updated, rays are cast from the right camera to its corresponding VMH (left). Red areas are carved. The result of this carving step is shown in the right image.

```
// Views traversal
for each camera do
    Get the camera position.
    Get the corresponding VMH.
    for each voxel in the VMH do
        Find the 3D Bresenham line between camera's position and voxel's center.
        Update each of these voxels' labels to transparent.
    end
end
```

Algorithm 2: Space carving algorithm.

At the end of the space carving step, the set of opaque voxels defines the volumetric model for our MVD data.

3.6 Marching Cubes

Finally, the merged surface mesh is then extracted from our binary-labelled volumetric representation using the Marching Cubes algorithm. Marching Cubes algorithm was originally introduced in[20] . It takes as input a regular volumetric data set. Such a data set has a scalar value assigned to each voxel vertex. During processing, each cube vertex that has a value less or equal than a predefined isovalue, is marked. All other vertices are left unmarked. Consequently the Marching Cubes algorithm outputs the triangular mesh connecting the marked vertices.

4. MULTI-VIEW TEXTURE MAPPING

The photoconsistency constraint is enforced by texturing the merged surface mesh extracted from Marching Cubes algorithm. Input texture images of the MVD data are used in order to texture the merged mesh. To this end, a new texture mapping algorithm is proposed based on a photoconsistency metric. The main steps of this algorithm are summarized in figure 6. First, visibility is determined for each triangle in the merged mesh. Second, each available texture is mapped on visible triangles of this merged mesh. Photometric projection error is computed for each triangle and for each input view and stored in error images. The best texture for each triangle is chosen as the one that minimizes a photoconsistency metric. Triangles not visible by any camera are textured with intermediate view. Synthesized views are produced by rendering the textured mesh, each triangle being textured with the best texture according to the photo consistency metric.

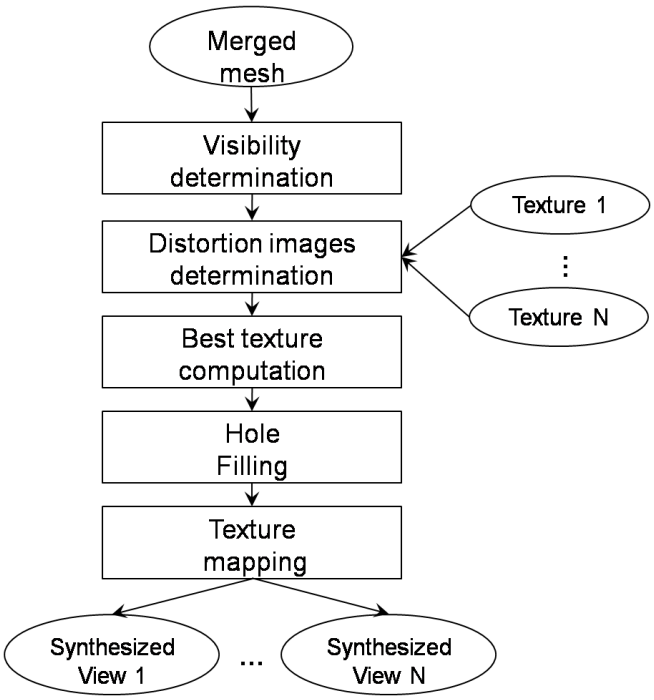


FIGURE 6. The overall framework of our texture mapping based on photoconsistency metric.

Given a set of texture images $\mathcal{I} = \{I_1, \ldots, I_n\}$ and a triangular mesh $\mathcal{M}$, the aim is to find for each mesh triangle $T \in \mathcal{M}$ the best texture image $\hat{I}_T \in \mathcal{I}$. Texturing the triangle T is formulated as an energy minimization problem. The best texture for T is found by minimizing a cost function formulated in terms of how much the triangle T is photoconsistent with the set of input images in which the triangle T is visible. Photo-consistency is estimated by projecting the textured mesh onto the views $\mathcal{V} = \{V_1, \ldots, V_n\}$ corresponding to input images $\mathcal{I}$.

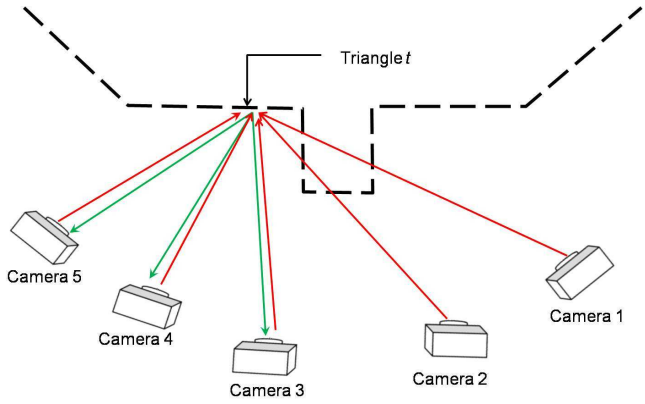


FIGURE 7. Texture mapping using the photoconsistency metric. For sake of simplicity, the surface mesh is shown with the black dotted line representing the mesh triangles. The triangle t is visible in cameras 3, 4, 5. For each texture $I_i \in \mathcal{I}$, the triangle t is textured with texture I_i (red arrow for texture mapping operation) and projected (green arrow for projection operation) onto the cameras 3, 4, and 5 - i.e. the set of cameras in which t is visible. The quadratic errors between the textured triangle and each of its projection on images I_3, I_4 and I_5 are computed. The error (i.e. the photoconsistency metric) of texturing the triangle t with the texture I_i is the sum of all these quadratic errors. The best texture is the one that minimizes the photoconsistency metric.

4.1 Visibility determination

From the merged mesh, visibility for each triangle is determined in order to perform texture mapping. The status of each mesh triangle is initialized to visible. Triangle visibility with respect to each camera is determined using OpenGL z-buffer. In the first pass, the input mesh is projected onto the current view, and the z-buffer is extracted. The second pass is dedicated to visibility determination using the computed z-buffer. Each mesh vertex is projected onto the current view and the depth component, denoted as $z_{\text{projected}}$, is checked against the pixel depth z_{buffer}. If the projected vertex is behind the pixel in the z-buffer, then this vertex is hidden, and thus the set of mesh triangles lying on this vertex are marked hidden. The pseudo-code is provided in algorithm 3.

```
//  Views traversal
for each view V_j do
    Initialize all triangles to visible.
    Project M onto V_j.
    Read z_buffer.
    // Mesh vertices traversal
    for each vertex v do
        Determine (q, l, z_projected) the projection of the vertex v(X, Y, Z) onto V_j.
        if z_buffer[q, l] > z_projected then
            v is a hidden vertex. Mark all the triangles lying on v as hidden.
        end
    end
end
```

Algorithm 3: Visibility determination

4.2 Distortion image computation

In order to determine the error of texturing a triangle with an input texture I_i, each available texture in then mapped on the merged mesh and projected onto each camera. Let $\mathcal{M}_{I_i \to V_j}$ be the projection onto the view V_j of the mesh $\mathcal{M}$ textured with the image I_i. For each available texture the following distortion image is computed :

$$D_{i,j} = \|\mathcal{M}_{I_i \to V_j} - I_j\|_2$$

```
// Textures traversal
for each texture I_i do
    // Views traversal
    for each view V_j do
        Compute M_{I_i → V_j} : projection of the mesh textured with texture I_i onto view V_j.
        D_ij = ‖ M_{I_i → V_j} − I_j ‖_2
    end
end
```

Algorithm 4: Compute distortion images.

4.3 Best texture determination

Among all the available textures the best texture image is chosen for each mesh triangle, using the distortion images. Let M_{T,V_j} be the projection of the triangle T onto the view V_j.

The photoconsistency metric measures the error of texturing the triangle T with the texture image I_i. $\forall i \in \{1, \ldots, n\}$ we compute :

$$\text{Error}_{T,I_i} = \sum_{\substack{j=1 \\ T \text{ visible in } V_j}}^{n} \left(\sum_{(q,l) \in M_{T,V_j}} D_{i,j}[q,l] \right) , \tag{4}$$

The best texture for a triangle T is then given by :

$$\hat{I}_T = \arg\min_{I_i \in \mathcal{I}} \text{Error}_{T,I_i} . \tag{5}$$

```
// Triangles traversal
for each triangle T do
    // Views traversal
    for each view V_j do
        Determine M_{T,V_j} the projection of the triangle T onto V_j where T is visible.
    end
    // Textures traversal
    for each texture I_i do
        Compute Error_{T,I_i}.
    end
    Determine Î_T.
end
```

Algorithm 5: Compute the best texture for each triangle.

4.4 Texture mapping

Finally, pictures are rendered. The 3D model is projected according to the virtual camera parameters supplied by the user. Then each triangle is textured using the texture coordinates determined during the above-mentioned texture mapping step.

4.5 Hole filling

Actually there are some triangles remaining whose all vertices are not visible simultaneously in the same view. These triangles can therefore obviously not be textured the usual way. In the current status of the modeling chain, these triangles are textured with texture coordinates from the central view, though it is not satisfying. Perspectives regarding this issue are twofold. First the final mesh could be warped for the number of such triangles be reduced. Second the remaining ones shall be inpainted in a 2D sense.

5. RESULTS

Results are presented for two sequences. The sequence *Breakdancers*, provided by Microsoft and shared in MPEG's former FTV group, presents 8 cameras arranged in the shape of an arc, with high-quality depth maps. The sequence *Balloons* was shot with a parallel setup. Rectified views and corresponding disparity maps were provided by Nagoya University and shared in MPEG's current standardization activities for 3DV technologies as a three-view test-sequence. The quality of *Balloons* depth maps is a bit lower than those of *Breakdancers*, but might be more realistic for practical applications. Both sequences present XGA resolution (1024x768).

5.1 Geometric modeling

The following figure illustrates the output surface of our geometric modeling scheme with a medium voxel resolution.

Geometric artifacts around sharp depth transitions are noticeable. At such resolution depth quantization artifacts also occur on plane surfaces. Both are expected to be lessened thanks to our texture mapping algorithm.

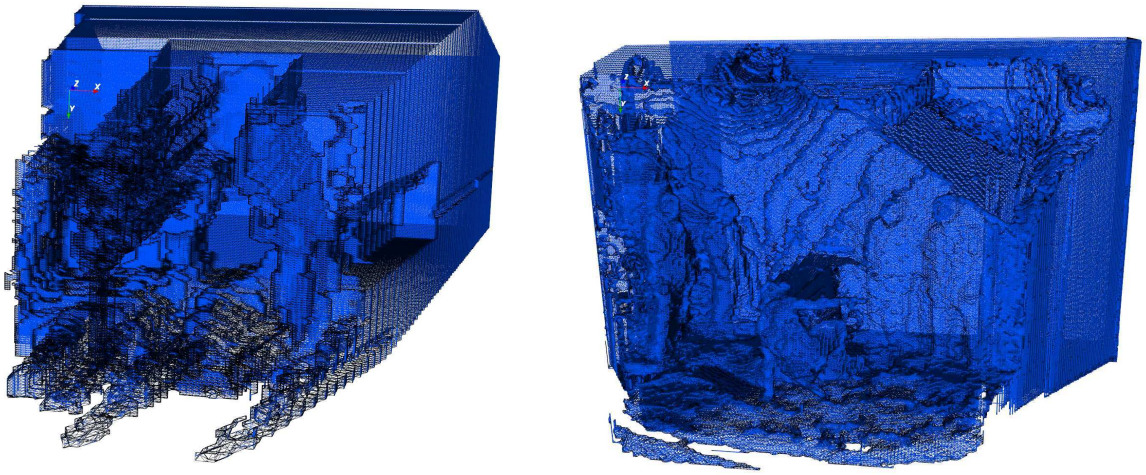


FIGURE 8. Results of our volumetric merging scheme using a volumetric resolution 250x250x250. Left : *Balloons* 3D mesh model. Right, *Breakdancers* 3D mesh model.

5.2 Texture mapping

Distortion computation : Distortion is computed between the rendered image and the original views using the PSNR metric. Note that distortion is computed only on triangles visible from at least one camera.

Camera	1	3	5
PSNR (dB)	28.9477	31.8392	30.9251

TABLE 1. Distortion of synthesized images for *balloons* sequences.

Camera	0	1	2	3	4	5	6	7
PSNR (dB)	28.6302	29.7552	31.2625	31.2188	31.7577	30.8564	29.863	29.2547

TABLE 2. Distortion of synthesized images for *breakdancers* sequences.

Though the distortion in terms of quadratic error is quite large, results are rather satisfying in terms of visual quality. Thanks to our texture mapping algorithm, our 3D modeling scheme is rather robust to input depth maps quality. Yet it is obvious that accurate registration and estimation helps in restoring ground truth in the geometric step.

6. CONCLUSION

In this paper we addressed the issue of 3D modeling natural video sequences from Multi-view Video plus Depth (MVD) material, without any assumption on the camera setup arrangement. A new geometrical modeling scheme was presented that merges the input depth maps into a single triangular mesh. To this end we designed a volumetric framework relying on an iterative Space Carving algorithm. Initial data are turned into voxels whose status (empty or non-empty) are updated according to their geometric consistency with the input depth maps. The frontier of the final volumetric model is turned back in a 3D mesh surface with Marching Cubes algorithm. A multi-view texturing scheme is also presented. This triangle-based algorithm assigns optimal textures coordinates to vertices of the final mesh. This is achieved by minimising color distortions between rendered pictures and the original views available at modeler side. Our 3D model consisting of both geometry and texture enables to synthesize any view around the initial cameras with few visual artifacts. As a perspective we plan to investigate the transmission issue, by reducing the coding cost of the geometrical model and formalizing the final texture signal to be delivered along.

FIGURE 9. Results of the proposed photoconsistency-based texture mapping on balloons sequences. Top left, reference texture image corresponding to camera 1. Top right, reference texture image corresponding to camera 5. Bottom left, synthesized image corresponding to camera 1. Bottom right, synthesized image corresponding to camera 5.

REFERENCES

[1] Fehn, C., Kauff, P., De Beeck, M., Ernst, F., Ijsselsteijn, W., Pollefeys, M., Van Gool, L., Ofek, E., and Sexton, I., "An evolutionary and optimised approach on 3d-tv," in [*Proc. of IBC*], **2**, 357–365 (2002).

[2] Merkle, P., Smolic, A., Muller, K., and Wiegand, T., "Multi-view video plus depth representation and coding," in [*Image Processing, 2007. ICIP 2007. IEEE International Conference on*], **1**, I–201, IEEE (2007).

[3] Balter, R., Gioia, P., and Morin, L., "Scalable and efficient video coding using 3-d modeling," *Multimedia, IEEE Transactions on* **8**(6), 1147–1155 (2006).

[4] Mueller, K., Smolic, A., Merkle, P., Kaspar, B., Eisert, P., and Wiegand, T., "3d reconstruction of natural scenes with view-adaptive multi-texturing," in [*3D Data Processing, Visualization and Transmission, 2004. 3DPVT 2004. Proceedings. 2nd International Symposium on*], 116–123, IEEE (2004).

[5] Waschbüsch, M., Würmlin, S., Cotting, D., Sadlo, F., and Gross, M., "Scalable 3d video of dynamic scenes," *The Visual Computer* **21**(8), 629–638 (2005).

[6] He, L., Shade, J., Gortler, S., and Szeliski, R., "Layered depth images," in [*Proceedings of the 25th annual conference on computer graphics and interactive techniques (SIGGRAPH 1998), July*], 19–24 (1998).

[7] Waschbüsch, M., Würmlin, S., and Gross, M., "3d video billboard clouds," in [*Computer Graphics Forum*], **26**(3), 561–569, Wiley Online Library (2007).

[8] Décoret, X., Durand, F., Sillion, F., and Dorsey, J., "Billboard clouds for extreme model simplification," in [*ACM Transactions on Graphics (TOG)*], **22**(3), 689–696, ACM (2003).

[9] Colleu, T., Pateux, S., Morin, L., and Labit, C., "A polygon soup representation for multiview coding," *Journal of Visual Communication and Image Representation* **21**(5-6), 561–576 (2010).

FIGURE 10. Results of the proposed photoconsistency-based texture mapping on Microsoft *breakdancers* sequences. Top left, reference texture image corresponding to camera 0. Top right, reference texture image corresponding to camera 7. Bottom left, synthesized image corresponding to camera 0. Bottom right, synthesized image corresponding to camera 7.

[10] McMillan, L. and Bishop, G., "Plenoptic modeling : An image-based rendering system," in [*Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*], 39–46, ACM (1995).

[11] Franco, J. and Boyer, E., "Efficient polyhedral modeling from silhouettes," *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **31**(3), 414–427 (2009).

[12] Dyer, C., "Volumetric scene reconstruction from multiple views," *Foundations of Image Understanding* , 469–489 (2001).

[13] Slabaugh, G., Culbertson, B., Malzbender, T., and Schafer, R., "A survey of methods for volumetric scene reconstruction from photographs," in [*International Workshop on Volume Graphics*], **2**(7), Citeseer (2001).

[14] Martin, W. and Aggarwal, J., "Volumetric descriptions of objects from multiple views," *Pattern Analysis and Machine Intelligence, IEEE Transactions on* (2), 150–158 (1983).

[15] Potmesil, M., "Generating octree models of 3d objects from their silhouettes in a sequence of images," *Computer Vision, Graphics, and Image Processing* **40**(1), 1–29 (1987).

[16] Kutulakos, K. and Seitz, S., "A theory of shape by space carving," *International Journal of Computer Vision* **38**(3), 199–218 (2000).

[17] Broadhurst, A., Drummond, T., and Cipolla, R., "A probabilistic framework for space carving," in [*Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*], **1**, 388–393, IEEE (2001).

[18] Vogiatzis, G., Torr, P., and Cipolla, R., "Multi-view stereo via volumetric graph-cuts," in [*Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*], **2**, 391–398, IEEE (2005).

[19] Curless, B. and Levoy, M., "A volumetric method for building complex models from range images," in [*Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*], 303–312, ACM (1996).
[20] Lorensen, W. and Cline, H., "Marching cubes : A high resolution 3d surface construction algorithm," *ACM Siggraph Computer Graphics* **21**(4), 163–169 (1987).

US009235922B2

(12) **United States Patent**
Alj et al.

(10) **Patent No.: US 9,235,922 B2**
(45) **Date of Patent: Jan. 12, 2016**

(54) **METHOD FOR MODELLING A 3D SCENE AND CORRESPONDING DEVICE**

(71) Applicant: **THOMSON LICENSING**, Issy de Moulineaux (FR)

(72) Inventors: **Youssef Alj**, Rennes (FR); **Guillaume Boisson**, Pleumeleuc (FR); **Philippe Bordes**, Laille (FR); **Luce Morin**, Rennes (FR); **Muriel Pressigout**, Rennes (FR)

(73) Assignee: **THOMSON LICENSING**, Issy les Moulineaux (FR)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 396 days.

(21) Appl. No.: **13/716,062**

(22) Filed: **Dec. 14, 2012**

(65) **Prior Publication Data**

US 2013/0187918 A1 Jul. 25, 2013

(30) **Foreign Application Priority Data**

Dec. 16, 2011 (FR) 11 61788

(51) **Int. Cl.**
G06T 15/08 (2011.01)
G06T 17/10 (2006.01)

(52) **U.S. Cl.**
CPC ***G06T 15/08*** (2013.01); ***G06T 17/10*** (2013.01)

(58) **Field of Classification Search**
CPC A61B 1/00193; H04N 2013/0081; H04N 13/0011; H04N 13/0282; H04N 19/00769; H04N 13/0018; G06T 7/0065; G06T 7/0075; G06T 2207/10012; G06T 15/205; G06T 2207/20228; G06T 15/00; G06T 2207/10028; G06T 2207/1001; G06T 15/08
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,952,204 B2 10/2005 Baumberg et al.
6,990,228 B1 1/2006 Wiles et al.
2011/0221861 A1* 9/2011 Jeon et al. 348/42

OTHER PUBLICATIONS

Publication Date: 2008 | Dec. 8-11, 2008 Document Title: Enforcing Image Consistency in Multiple 3-D Object ModellingAuthors: Grum, M. Bors, A.G.Organization: Dept. of Comput. Sci., Univ. of York, YorkSource: ICPR 2008 19th International Conference on Pattern RecognitionPublisher: IEEE.

(Continued)

Primary Examiner — Haixia Du
(74) *Attorney, Agent, or Firm* — Tutunjian & Bitetto, P.C.

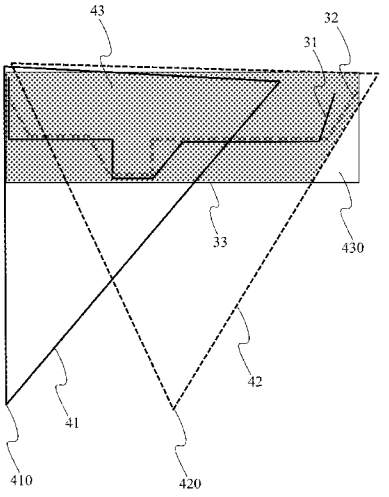
(57) **ABSTRACT**

The invention relates to a method for modeling a scene from a plurality of maps representative of depth, each map representative of depth being associated with a view of the said scene according to a particular viewpoint, To optimize the fidelity and precision of the scene modeling, the method comprises the following steps:

- generating, for each map representative of depth, a surface representative of the said depth map in world space;
- estimating a volume bounding all surfaces generated, the said bounding volume being discretised into a plurality of voxels;
- associating attributes representative of transparency with the said voxels of the bounding volume; and
- generating a resulting surface according to the attributes representative of transparency associated with the said voxels.

The invention also relates to a device for processing the data representative of the corresponding depth.

23 Claims, 8 Drawing Sheets



(56) **References Cited**

OTHER PUBLICATIONS

Publication Date: 2008 | Jan. 22-25, 2008Document Title: Exact Visual Hull From Marching CubesAuthors: Chen Liang Wong, K.-Y. K.Department of Computer Science, The University of Hong Kong, Pokfulam, Hong Kongcliang@cs.hku.kh, kykwong@cs.hku.hkOrganization: Dept. of Comput. Sci., Univ. of Hong Kong, Hong KongSource: VISAPP 2008. Proceedings of the Third International Conference on Computer Vision Theory and ApplicationsPublisher: INSTICC—Institute for Systems and Technologies of Information, Control, and.

ProjectiveVisual HullsPublication Date: Aug. 2007Authors:Svetlana Lazebnik1 (slazebni@uiuc.edu)Yasutaka Furukawa1 (yfurukaw@uiuc.edu)Jean Ponce1,2 (jponce@uiuc.edu)1 Department of Computer Science and Beckman InstituteUniversity of Illinois, Urbana, IL 61801, USA2 Département d'InformatiqueEcole Normale Sup'erieure, Paris, France.

System for Reconstruction of Three-Dimensional Micro Objects From Multiple Photographic Imagesauthors: Atsushi, K. Sueyasu, H. Funayama, Y. Maekawa, T.Organization: Dept. of Mech. Eng., Yokohama Nat. Univ., Yokohama | Ricoh Co. Ltd. Technol. Center, 2011.

Aggraval et al—Volumetric descriptions of objects from mulitple views—IEEE Transactions on Pattern Analysis and Machine Intelligence 5(2)-150 158—Mar. 1983.

Edelsbrunner et al—Three-dimensional alpha shapes In Workshopon Volume Visualization pp. 75-105 Oct. 1992.

Kaazdhan et al_Poisson surface reconstruction—In Proc SGP pp. 67-70-2006.

Kutulakos et al—A theory of shape by space carving. International Journal of Computer Vision 38(3) 199-218-2000.

Hyojin Kim et al. "GPU Based Scalable Volumetric Recosntruction for Multi View Stereo" IVCNS 2011 (Nov. 29, 2011).

Grosso Etal "3D Object Reconstruction Using Stereo and Motion" IEEE Transactions on Systems, Man and Cybernetics, IEEC Inc, New York, US, vol. 19, N° 6, Nov. 1, 1989, pp. 1465-1476.

Alj Y et al. "Space Carving MVD Sequences for Modeling Natural 3D Scenes" SPIE Proceedings: Three Dimensional Image Processing (3DIP) and Application II, The International Society for Opitical Engineering—SPI, Bellingham, 2012.

Search Report Dated Sep. 21, 2012.

* cited by examiner

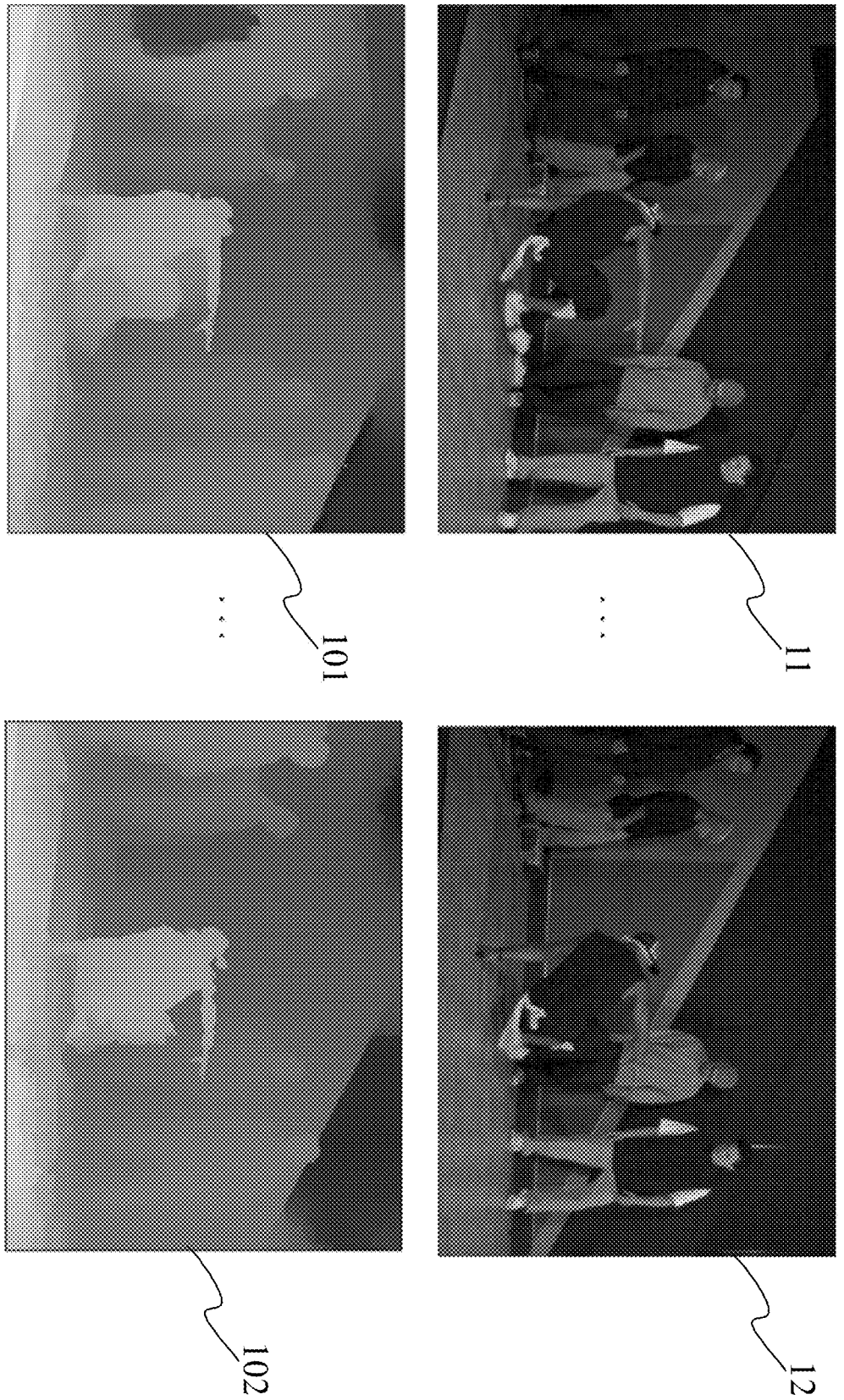


Fig 1

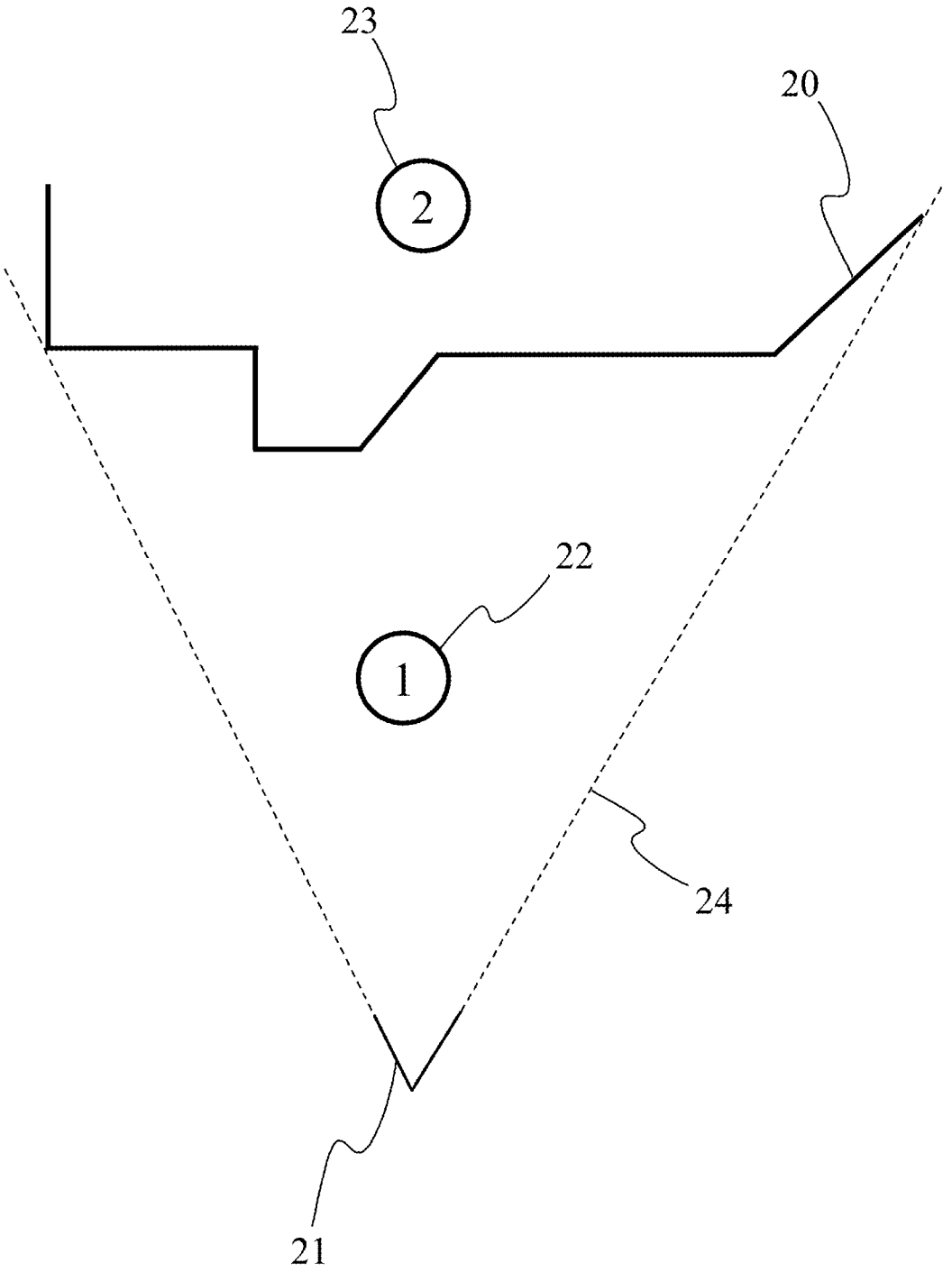


Fig 2

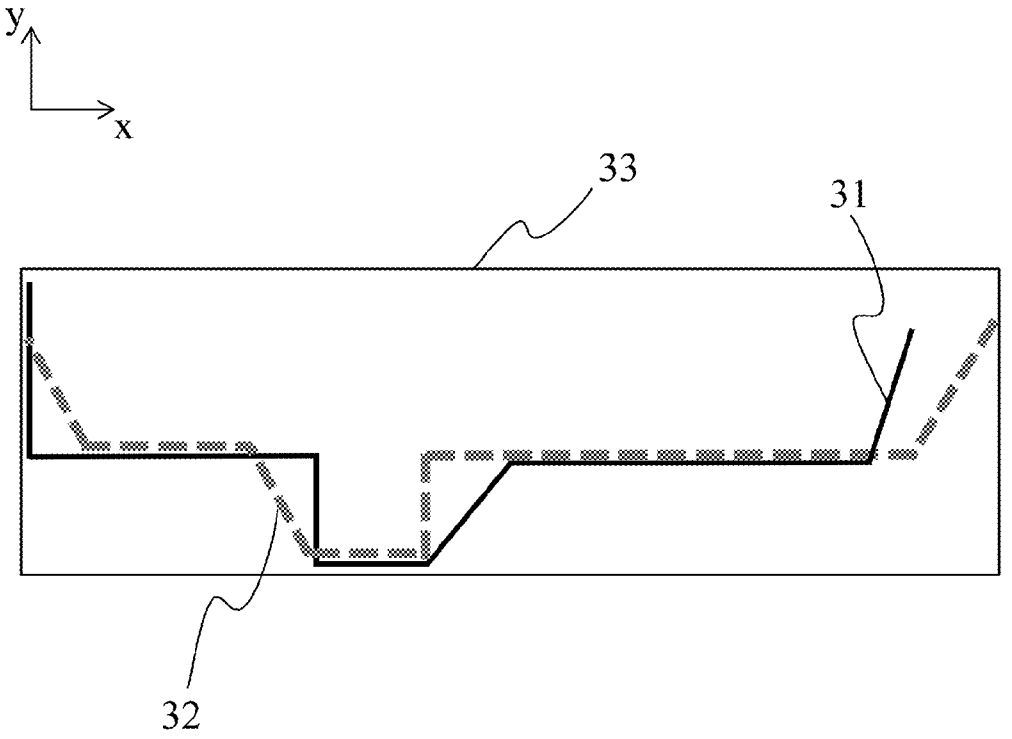


Fig 3

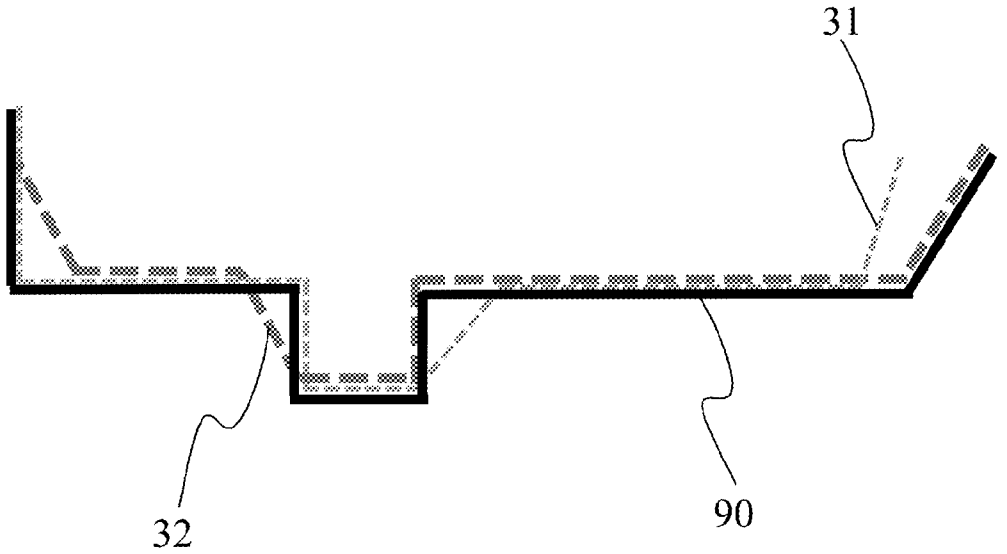


Fig 9

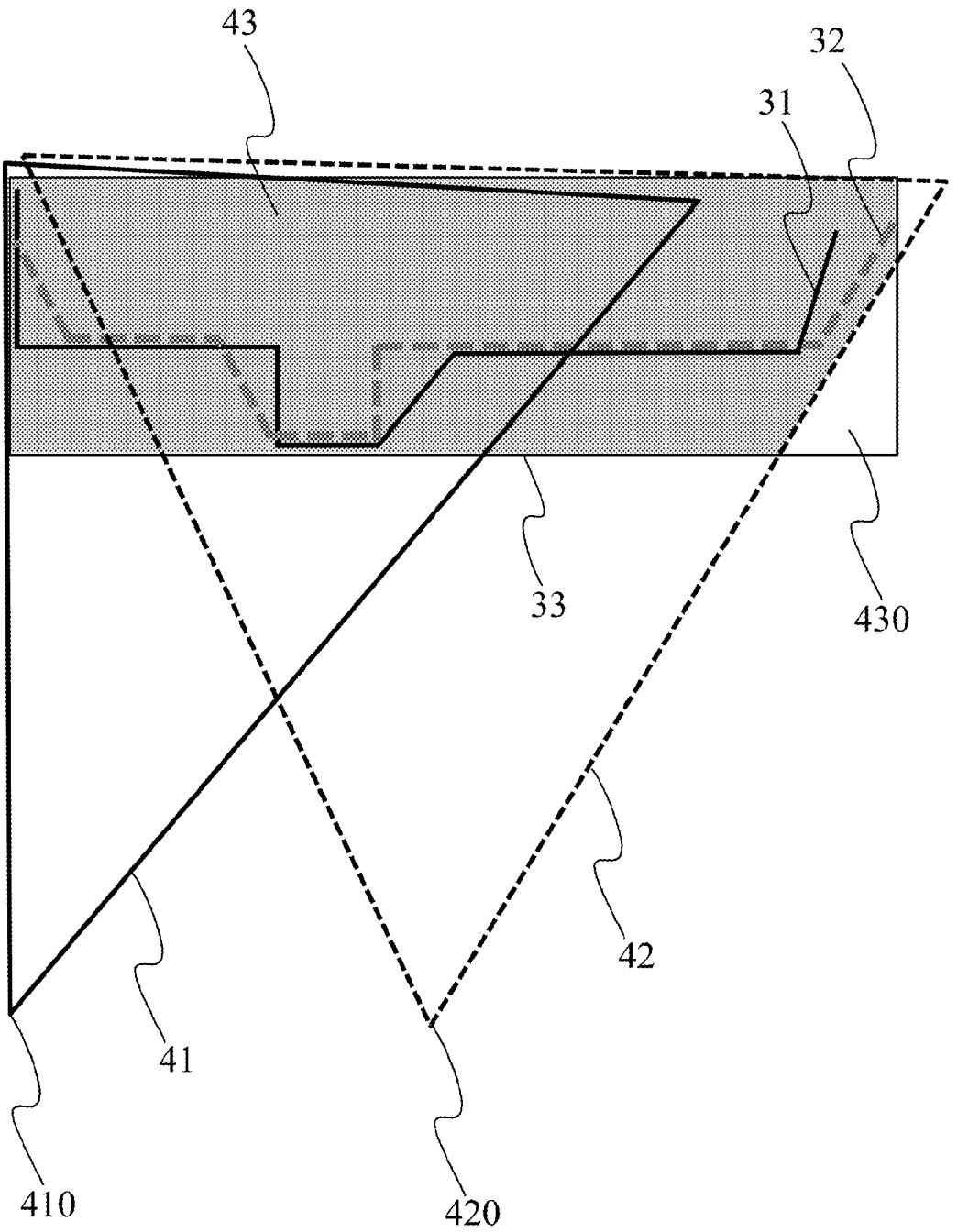


Fig 4

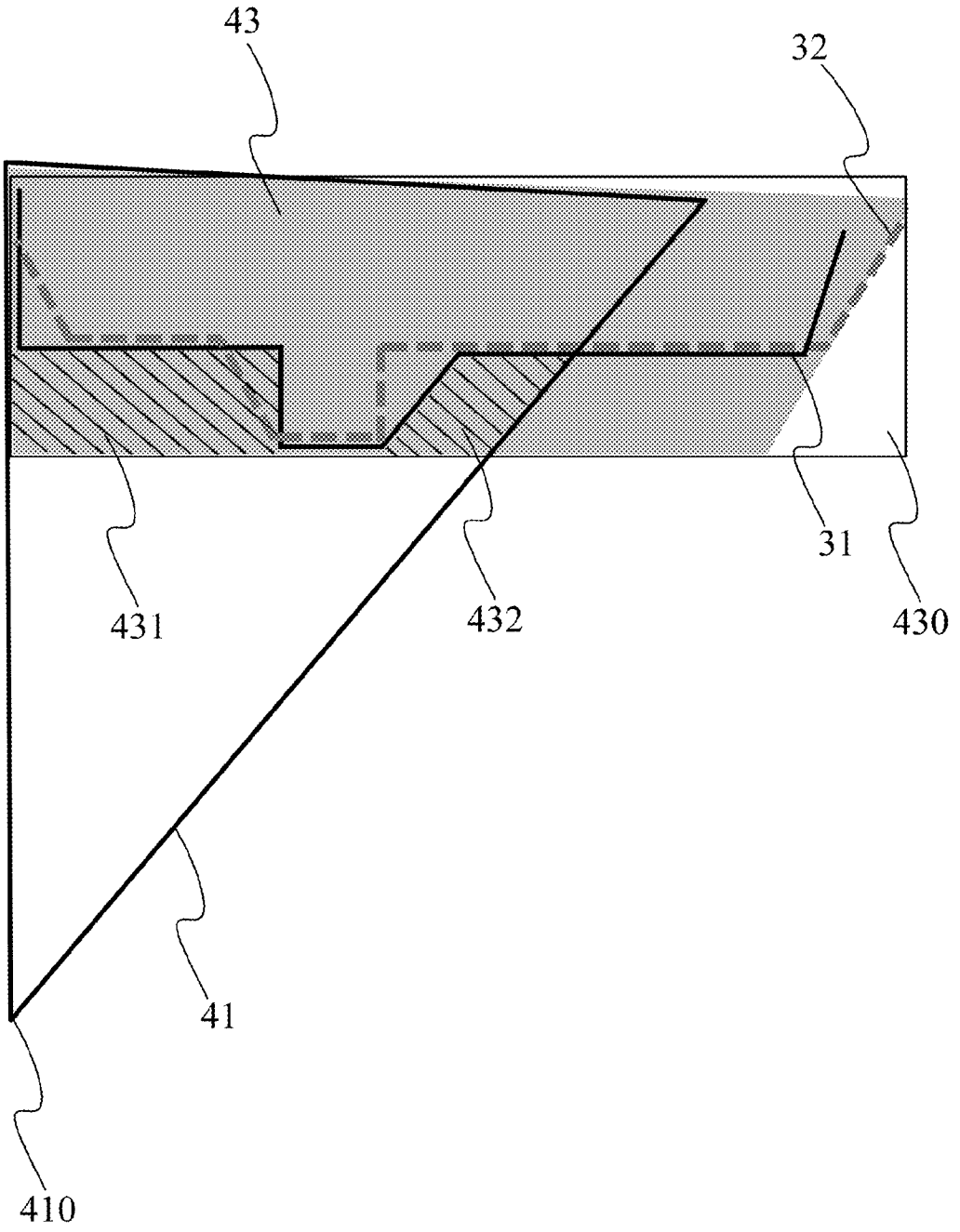


Fig 5

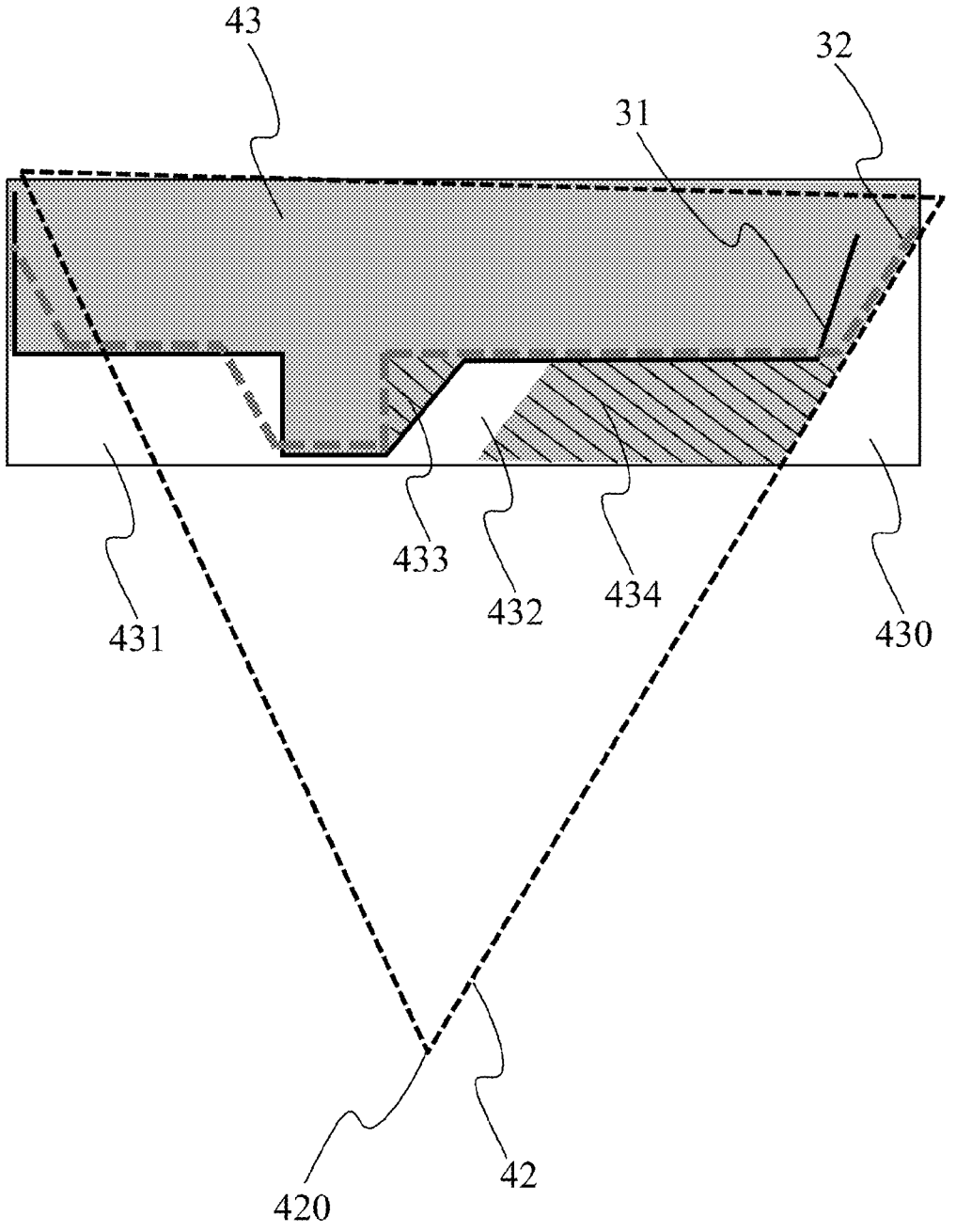


Fig 6

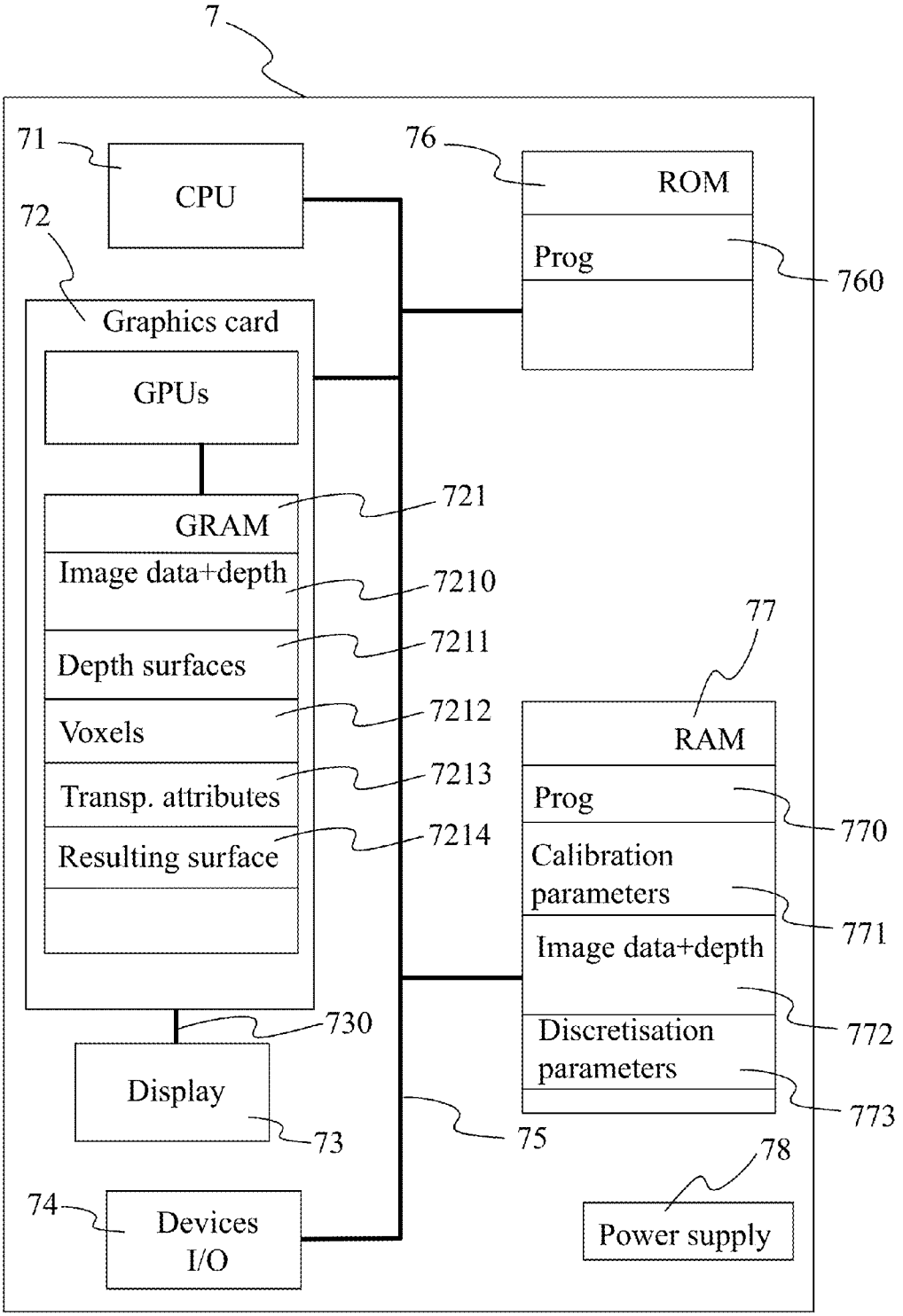

7
71
CPU
72
Graphics card
GPUs
GRAM
Image data+depth
Depth surfaces
Voxels
Transp. attributes
Resulting surface
721
7210
7211
7212
7213
7214
730
Display
73
74
Devices I/O
75
76
ROM
Prog
760
77
RAM
Prog
770
Calibration parameters
771
Image data+depth
772
Discretisation parameters
773
78
Power supply

Fig 7

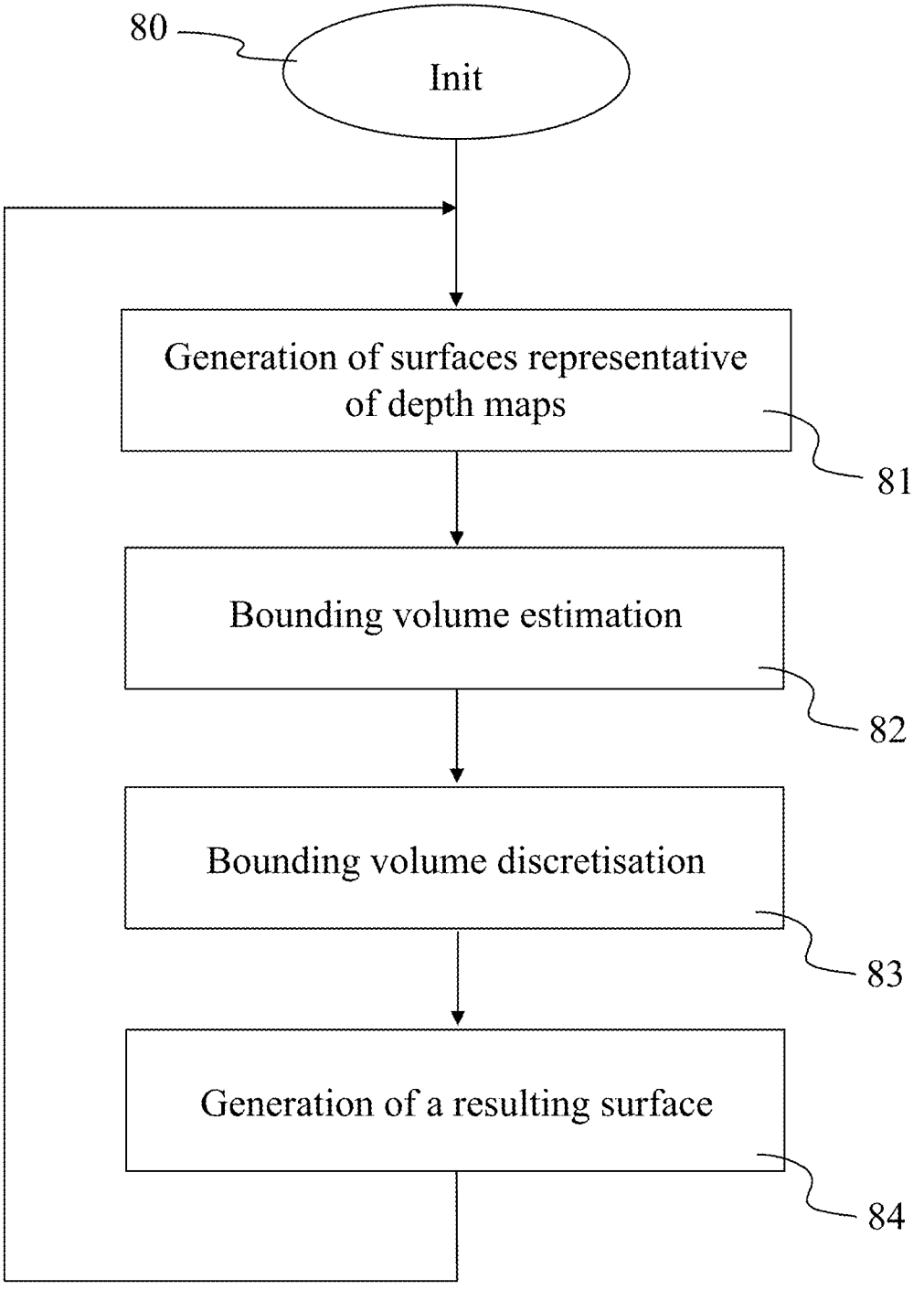

80
Init
Generation of surfaces representative of depth maps
81
Bounding volume estimation
82
Bounding volume discretisation
83
Generation of a resulting surface
84

Fig 8

METHOD FOR MODELLING A 3D SCENE AND CORRESPONDING DEVICE

This application claims the benefit, under 35 U.S.C. §119 of French Patent Application 1161788, filed Dec. 16, 2011.

1. DOMAIN OF THE INVENTION

The invention relates to the domain of image or video processing and more specifically to data processing representative of depth; for example depth maps or disparity maps associated with images of a video stream. The invention also relates to the domain of modelling of the geometry of a real scene using the depth data associated with images representative of the scene from several viewpoints.

2. PRIOR ART

According to the prior art, several methods of modelling a real scene in three dimensions (3D) are known. Among these methods, the technique called "Visual Hull" is used to reconstruct a 3D real scene from images of the scene captured by a set of cameras from several viewpoints. A silhouette of the scene is calculated for each image. A volumetric intersection of the de-projections of these silhouettes in world space enables a 3D model of the captured scene to be reconstructed. One disadvantage of this method is that it induces modelling errors of concave areas (see for example the document entitled "*Volumetric descriptions of objects from multiple views*" by W. N. Martin and J. K. Aggarwal published in *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 5(2):150-158, March 1983).

Another technique called "Space carving" is used to model a real scene captured by multiple cameras from several viewpoints. According to this method, a volume bounding the objects in the captured scene is defined then discretised into a plurality of voxels. The photo-consistency of each voxel of the bounding volume is then defined by projecting each voxel onto the image planes associated with the cameras having captured the scene images, a voxel being photo-consistent if its projection is identical (that is to say that the variation of the colour associated with the projections of the voxel is less than a threshold value) on each image plane. To reconstruct a 3D model of the scene, the voxels which are not photo-consistent are eliminated (see the document entitled "*A theory of shape by space carving*" by K. N. Kutulakos and S. M Seitz published in *International Journal of Computer Vision,* 38 (3), 199-219 (2000)). The method known as "Space carving" presents several disadvantages. One of the disadvantages arises from the use of a threshold value to determine the photo-consistency of voxels, photo-consistency estimation errors being induced by an inappropriate threshold value. Another problem emerges when the captured scene comprises surfaces with significant light variations. Such areas, over-lighted or having significant light variations, induce erroneous photo-consistency determinations and therefore the elimination of incorrect voxels.

3. SUMMARY OF THE INVENTION

The purpose of the invention is to overcome at least one of these disadvantages of the prior art.

More specifically, the purpose of the invention is notably to optimise the scene modelling fidelity and accuracy.

The invention relates to a method of modelling a scene from a plurality of maps representative of depth, each map representative of depth being associated with a view of the scene according to a particular viewpoint. The method comprises the following steps:

- generating, for each map representative of depth of the plurality of maps, a surface representative of the map representative of depth in world space;
- estimating a volume bounding the surfaces generated in world space, the bounding volume being discretized into a plurality of voxels, an attribute representative of transparency being associated with the voxels of the bounding volume;
- generating, for each surface, a plurality of rays originating from a viewpoint associated with the generated surface and having as end point the intersection with the generated surface;
- running through the plurality of generated rays, the value of the attribute representative of transparency being changed for the voxels of the bounding volume having an intersection with one of the generated rays; and
- generating data representative of a resulting surface according to the attributes representative of transparency associated with the voxels.

Advantageously, the method comprises, for each surface generated and representative of a depth map associated with a viewpoint, an association of the voxels of the bounding volume with the generated surface to define the generated surface in accordance with a criterion of proximity of the voxels relative to the generated surface;

According to a particular characteristic, an attribute representative of transparency comprises two states, a first state corresponding to a transparent state of the voxel associated with the attribute, a second state corresponding to an opaque state of the voxel associated with the attribute, a voxel being in the transparent state when it has an intersection with one ray of the plurality of rays.

Advantageously, an attribute representative of transparency is determined according of the position of the voxel associated with the attribute with respect to a viewpoint and to the surface representative of a depth map associated with the viewpoint.

According to a specific characteristic, the attribute comprises two states, a first state corresponding to a transparent state of the voxel associated with the attribute, a second state corresponding to an opaque state of the voxel associated with the attribute, a voxel being in the transparent state when it is positioned between a viewpoint and the surface representative of the depth map associated with the viewpoint in the world space.

Advantageously, the position of the voxel is determined by estimating the intersection between the voxel and a ray launched from the viewpoint.

According to a particular characteristic, the surface representative of the depth map in the world space is generated using parameters representative of calibration associated with at least one camera used to capture the views of the scene associated with the plurality of depth maps.

According to another characteristic, the plurality of depth maps is obtained from a multi-view video stream.

Advantageously, the video stream is an MVD stream "Multiview Video plus Depth".

According to a specific characteristic, the method comprises a step of transmitting depth information representative of the resulting surface.

The invention also relates to a device for processing data representative of a plurality of maps representative of depth, each map representative of depth being associated with a view

of a scene according to a particular viewpoint, the device comprising at least a processor configured for:

- generating, for each map representative of depth of the plurality of maps, a surface representative of the map representative of depth in world space;
- estimating a volume bounding the surfaces generated in world space, the bounding volume being discretised into a plurality of voxels, an attribute representative of transparency being associated with the voxels of the bounding volume;
- generating, for each surface, a plurality of rays originating from a viewpoint associated with the generated surface and having as end point the intersection with the generated surface;
- running through the plurality of generated rays, the value of the attribute representative of transparency being changed for the voxels of the bounding volume having an intersection with one of the generated rays; and
- generating data representative of a resulting surface according to the attributes representative of transparency associated with the voxels.

4. LIST OF FIGURES

The invention will be better understood, and other specific features and advantages will emerge upon reading the following description, the description making reference to the annexed drawings wherein:

FIG. **1** illustrates a plurality of images representative of a scene and depth maps associated with this plurality of images, according to an example of a particular implementation of the invention;

FIG. **2** illustrates a surface representative of a depth map in FIG. **1**, according to a particular embodiment of the invention;

FIG. **3** illustrates a volume bounding a plurality of surfaces representative of the depth maps in FIG. **1**, according to a particular embodiment of the invention;

FIGS. **4**, **5** and **6** illustrate steps of the method for modelling the scene represented by the plurality of images in FIG. **1**, according to a particular embodiment of the invention;

FIG. **7** diagrammatically shows the structure of a unit for processing data representative of the depth maps in FIG. **1**, according to a particular embodiment of the invention;

FIG. **8** illustrates a method for modelling a scene from the depth maps in FIG. **1**, implemented on a processing unit in FIG. **7**, according to a particular embodiment of the invention;

FIG. **9** illustrates a surface resulting from the processing performed on the surfaces of the bounding volume in FIG. **3** according to the steps of the method shown in FIGS. **4**, **5** and **6**, according to a particular embodiment of the invention.

5. DETAILED DESCRIPTION OF THE EMBODIMENTS OF THE INVENTION

The invention will be described with reference to a first particular embodiment of a method for modelling a scene. The scene is advantageously represented according to several viewpoints in several images, each image being associated with information representative of depths. Information on depths is stored for example in depth maps or disparity maps. A particular viewpoint is associated with each map comprising depth information associated with the image captured according to the viewpoint considered. For each map comprising information representative of depth, a surface is generated in world space (that is to say in the space of the scene), the surface consequently representing the map comprising information representative of depth in world space according to the viewpoint associated with the map considered. A bounding volume is defined, which comprises all the generated and representative surfaces of maps comprising information representative of depth according to the different viewpoints. The bounding volume is then discretised, that is to say divided into a plurality of voxels, an attribute representative of transparency being associated with each voxel. A surface resulting from the fusion of all previously generated surfaces is then determined from the attributes representative of transparency associated with the voxels of the bounding volume.

Taking into account a geometric criterion (that is to say depth information) to determine a surface resulting from the fusion of several surfaces representing the depth information associated with the scene to be modelled according to several viewpoints has the advantage of overcoming the problems generated by taking into account a photometric criterion (that is to say the colour information associated with pixels of images representing the scene according to different viewpoints), while ensuring high fidelity in the modelling of the scene taking into consideration the concave and convex parts of the scene.

FIG. **1** illustrates a plurality of images **11**, **12** representative of a scene and the maps representative of depth **101**, **102** associated with each of the images **11**, **12**, according to an example of a particular and non-restrictive implementation of the invention. Images **11** and **12** represent the same scene from two different viewpoints. The video content of image **12** corresponds to the video content of image **11** shifted according to a horizontal axis towards the right. The scene represented in images **11** and **12** is advantageously captured by two acquisition devices, for example two cameras spatially offset in relation to one another. Each image **11**, **12** is associated with data representative of depth, for example a depth map (e.g. "z-buffer" type) or a disparity map. For the rest of the description we will make no distinction between a map representative of depth and a depth map, it being understood that it can concern any data structure representative of depth or disparity information. According to the example illustrated with respect to FIG. **1**, a depth map **101** is associated with image **11** and a depth map **102** is associated with image **12**. The depth map **101** (respectively **102**) represents in gray levels the depth associated with each pixel of the image **11** (respectively **12**), the closest pixels of the image (that is to say having the lowest depth values, the reference being the viewpoint of the image or the acquisition device having being used to capture the image) being the brightest and the most distant pixels of the image (that is to say having the highest depth values) being the darkest. Advantageously, the depth information associated with the pixels of images **11** and **12** are captured by means of suitable sensors, for example by means of infra-red sensors associated with an infrared transmitter (Kinect® type system) or by means of a system comprising a laser transmitter and an associated sensor to set the travel time of a laser ray emitted to the scene captured and reflected by the same scene, the travel time being representative of the distance travelled and therefore the depth associated with the scene point which reflected the laser ray.

According to a variant, a disparity map is associated with each image **11**, **12**. According to this variant, the disparity map associated with the image **11** is representative of the disparity between image **11** and image **12** and the disparity map associated with image **12** is representative of the disparity between image **12** and image **11**. Each disparity map is advantageously estimated by comparison and pairing of the

pixels of image **11** (respectively **12**) to the pixels of image **12** (respectively **11**). The disparity associated with a pixel of an image **11** (respectively **12**) advantageously corresponds to the pixel distance between this pixel of image **11** (respectively **12**) and the corresponding (or paired) pixel of image **12** (respectively **11**), that is to say, the pixel of the image **12** (respectively **11**) having video information (that is to say, colour information) identical or similar to that of the pixel of image **11** (respectively **12**).

According to another variant, the depth map **101**, **102** associated with each image **11**, **12** is deduced from the disparity map associated with each image **11**, **12**.

The images **11** and **12** belong to a multi-view video stream comprising a sequence of several images representative of the same scene. Advantageously, the multi-view video stream is of MVD type (Multiview Video plus Depth).

According to a variant, the colour and depth information associated with pixels of images **11** and **12** are stored in memories in the form of RGBa information, the RGB channels (Red, Green, Blue) being used to store the colour information associated with each pixel (e.g. 8, 10 or 12 bits per channel) and the channel a being used to store the depth (or disparity) information, for example on 8, 10 or 12 bits.

Advantageously, the intrinsic and extrinsic parameters of the acquisition devices used to acquire the plurality of images are known and for example stored in a memory. The intrinsic and extrinsic parameters correspond to the calibration parameters of acquisition devices. The intrinsic parameters comprise for example the focal length, magnification factors of the image, the projection coordinates of the optical centre of the acquisition device on the image plane and/or a parameter translating the potential non-orthogonality of lines and columns of photosensitive cells which compose the sensor of the acquisition device. The extrinsic parameters comprise for example the rotation matrix used to shift from the world space reference to the image space reference (and vice versa) and/or the components of the translation vector used to move from the world space reference to the image space reference (and vice versa).

Naturally, the number of images representative of a scene is not limited to 2 images but extends to any number greater than 2, e.g. 3, 4, 5, 10, 20, 50, 100 images, data representative of depth being associated with each image. Advantageously, each frame is captured by a particular acquisition device according to a particular viewpoint, the acquisition devices being spatially offset in relation to one another. According to a variant, only one pair of left and right images is captured using two acquisition devices, the other images of the plurality representative of the captured scene being calculated from the pair of left and right images by disparity-compensated interpolation. According to this variant, each image is also representative of the same scene according to a particular viewpoint and data representative of depth (for example a depth map or disparity map) is associated with each image, whether captured or interpolated.

FIG. **2** illustrates a surface **20** representative of a depth map **101**, **102**, according to a particular and non-restrictive embodiment of the invention. For obvious practical reasons, the surface **20** illustrated is a 2D representation of the surface representative of a depth map of an image, for example a sectional top view of the surface. The surface **20** also represents the surface of the visible points or elements of an image as viewed from a viewpoint **21**. We refer to points when the image is represented by a plurality of points (or pixels). We refer to elements when the image is represented by elements, for example the elements of a mesh (e.g. triangles or parallelepipeds). According to a variant, an element corresponds to a voxel and the surface is thus defined by the limit (or one face) of adjacent voxels.

The area (**1**) **22** delimited on the one hand by the view cone having for its vertex the viewpoint **21** and on the other hand the surface **20** corresponds to the geometric inconsistency area, that is to say the area comprising the geometrically inconsistent points (or voxels). By geometrically inconsistent points/voxels is understood the set of points or voxels that do not belong to an object of the 3D-modelled scene (the scene being represented in one of the images **11**, **12**). Such a point or voxel not belonging to an object of the scene is also called transparent. A point/voxel is called geometrically incoherent if the distance which separates it from the viewpoint **21** according to a ray originating from the viewpoint and passing through the point/voxel in question is less than the distance from the viewpoint **21** of the point of the surface **20** encountered by the ray passing through the point/voxel in question. According to a variant, any point/voxel belonging to a ray originating from the viewpoint **21** and having as an end point a point of the surface **20**, is called geometrically inconsistent.

The area (**2**) **23** comprises all points/voxels placed behind the surface **20**, if we are positioned at the viewpoint **21**, corresponds to an area of geometrical consistency, that is to say an area comprising the geometrically consistent points/voxels. By geometrically consistent points/voxels is understood the set of points or voxels belonging to an object in the 3D-modelled scene (the scene being represented in one of the images **11**, **12**). Such a point or voxel not belonging to an object of the scene is also called opaque. A point/voxel is called geometrically consistent if the distance which separates it from the viewpoint **21**, according to a ray originating from the viewpoint and passing through the point/voxel in question, is greater than the distance separating the viewpoint **21** from the point of the surface **20** encountered by the ray passing through the point/voxel in question. According to a variant, any point/voxel not belonging to a ray originating from the viewpoint **21** and having as an end point a point on the surface **20** is called geometrically consistent.

FIG. **3** illustrates a volume **33** bounding a plurality of surfaces **31**, **32** representative of the depth maps **101**, **102**, according to a particular and non-restrictive embodiment of the invention. The surface **31** shown in solid line is for example representative of the depth map **101** according to the viewpoint associated with the image **11**, the depth map **101** itself being associated with the image **11**. The surface **31** thus represents the set of points or elements of the scene represented by image **11** visible from the viewpoint associated with image **11**. The surface **32** shown in dotted line is for example representative of the depth map **102** according to the viewpoint associated with image **12**, the depth map **102** itself being associated with image **12**. The surface **32** thus represents the set of points or elements of the scene represented by image **12** visible from the viewpoint associated with image **12**. The surfaces **31** and **32** are two-dimensional representations of three-dimensional surfaces. The illustration in FIG. **3** of these surfaces **31** and **32** corresponds for example to a sectional top-view of 3D surfaces. The volume **33** (called bounding volume in the following description) is defined so that it fully comprises the surfaces **31** and **32**, the limits of the bounding volume being determined by points positioned at the end points of the surfaces **31** and **32**, that is to say the points of the surfaces **31** or **32** having the most extreme x, y and z coordinates (the lowest and highest values for each axis x, y and z). Once defined, the bounding volume **33** is discretised into a plurality of voxels, the voxels being advantageously all of the same dimensions (a voxel corresponds for

example to a cube or a rectangular parallelepiped). The discretisation parameters (such as the number of voxels per volume unit or the dimensions of a voxel in x, y and z) are advantageously predefined. According to a variant, the discretisation parameters are adjustable by a user, for example via a user interface. Advantageously, an attribute representative of transparency is associated with each voxel of the bounding volume **33**. The attribute representative of transparency comprises two distinct states, namely the transparent state and the opaque state. The opaque state is advantageously associated with each of the voxels of the bounding volume after the discretisation into voxels of the bounding volume.

The surfaces **31** and **32** are respectively representative of the depth maps **101** and **102** in world space, the depth maps **101** and **102** comprising information on depth in the image plane. By world space is understood the reference of the scene captured by the acquisition devices and represented by the images **11** and **12**, which correspond to a plurality of 2D views of a 3D scene. The generation of a surface **31** or **32** in the world space from a depth map **101** or **102** as represented in the image space is based on intrinsic and extrinsic parameters of acquisition devices that were used to capture the images **11** or **12** representing the scene that we seek to model.

FIGS. **4**, **5**, **6** and **9** illustrate the steps of a method for modelling the scene represented by the plurality of images **11**, **12**, according to a particular and non-restrictive embodiment of the invention.

FIG. **4** illustrates a first step of the method for modelling the scene based on the depth maps **101** and **102** associated with the images **11** and **12** representing the captured scene. In FIG. **4** we find the bounding volume **33**, which surrounds surfaces **31** and **32**, surface **31** being representative of the depth map **101** associated with image **11** and surface **32** being representative of the depth map **102** associated with image **12**. A first view cone **41** shown in solid line is defined, the first view cone **41** corresponding to what is seen from a first viewpoint **410** corresponding to the viewpoint from which the image **11** was captured. The first view cone **41** is consequently associated with the surface **31** representative of the depth map **101**. A second view cone **42** shown in dotted line is defined, the second view cone **42** corresponding to what is seen from a second viewpoint **420** corresponding to the viewpoint from which the image **12** was captured. The second view cone **42** is consequently associated with the surface **32** representative of the depth map **102**. Once the view cones **41**, **42** are defined, the intersection between these view cones **41**, **42** on the one hand and the bounding volume **33** on the other hand is determined. This intersection, which corresponds to a volume, is illustrated by an area **43** in gray on FIG. **4**. In other words, the volume of intersection **43** corresponds to the union of view cones **41**, **42** inside the bounding volume **33**. From the intersection volume **43**, we determine the voxels belonging to the bounding volume **33** but not to the intersection volume **43**. By voxels of the bounding volume **33** not belonging to the intersection volume **43** is understood those voxels whose centre, defined for example by its x, y, and z coordinates, does not belong to the intersection volume **43**. This plurality of voxels not belonging to the intersection volume **43** is represented by a blank area **430** corresponding to a volume inside the bounding volume **33**. Advantageously, the attribute representative of transparency associated with each of the voxels of the area **430** is modified and each voxel of the area **430** passes to the transparent state, the voxels belonging to the intersection volume **43** remaining in their initial state, that is to say in the opaque state as defined with respect to FIG. **3**.

FIG. **5** illustrates a second step of the method for modelling the scene based on the depth maps **101**, **102** associated with the images **11**, **12** representing the captured scene. During this second step, the first view cone **41** associated with the surface **31** representative of the depth map **101** is selected. A plurality of rays is launched inside the first view cone **41**, the launched rays originating from the first viewpoint **410**. Advantageously, the rays launched from the first viewpoint **410** have as an end point the intersection with the surface **31**. The intersection with the surface **31** corresponds advantageously to a voxel, a series of voxels being associated with the surface **31** according to a criterion of proximity of the voxels in relation to the surface **31**. The proximity of the voxels to the surface **31** is advantageously estimated by measuring the distance between an element of the surface **31** and the centre of the voxels surrounding this element. The selection of voxels associated with the surface **31** is advantageously performed using a method of filling by diffusion, for example the method called "scan-line fill", the method called "seed fill" or the method called "flood fill". Running through each of the launched rays, we determine the voxels of the bounding volume **33** having an intersection with these rays. The attribute representative of transparency associated with these voxels having an intersection with the launched rays (initialised in opaque state) is modified and each of these voxels passes to the transparent state. This means that the attributes representative of transparency associated with the voxels of the bounding volume are thus determined according to the result of the intersection estimate between the voxels and launched rays. The voxels having an intersection with these launched rays and whose state passes to transparent belong to the areas **431** and **432** showing as hatched in FIG. **5**.

According to a variant, the rays launched from the first viewpoint **410** have as an end point the second intersection with the bounding volume **33**, that is to say, the intersection with the face of the bounding volume farthest from the first viewpoint. For each ray launched and for each voxel belonging to one of these rays, the distance between the origin (that is to say the first viewpoint **410**) and the voxel considered is determined and compared to the distance separating the origin of the intersection point between the surface **31** and the ray considered. If the distance to the voxel is less than the distance to the surface **31**, then the voxel passes to the transparent state. In the contrary case, if the distance to the voxel is greater than the distance to the surface **31**, then the voxel remains in the opaque state. According to this variant the same voxels pass to the transparent state and belong to the areas **431** and **432** illustrated as hatched in FIG. **5**. This second step is used to define the geometrically inconsistent voxels according to the first viewpoint **410**, that is to say the voxels that do not belong to an object of the scene represented by image **11** associated with the first viewpoint **410**.

FIG. **6** illustrates a third step of the method for modelling the scene based on the depth maps **101**, **102** associated with the images **11**, **12** representing the captured scene. The third step is similar to the second step illustrated with respect to FIG. **5**, the difference being that it is applied to the second view cone **42**. During this third step, the second view cone **42** associated with the surface **32** representative of the depth map **102** is selected. A plurality of rays is launched inside the second view cone **42**, the launched rays originating from the second viewpoint **420**. Advantageously, the rays launched from the second viewpoint **420** have as an end point the intersection with the surface **32**. Running through each of the launched rays, we determine the voxels of the bounding volume **33** having an intersection with these rays. The attribute representative of transparency associated with these voxels having an intersection with the launched rays (initialised in opaque state) is modified and each of these voxels passes to the transparent state. The voxels having an intersection with these launched rays and whose state passes to the transparent state belong to the areas **433** and **434** which appear as hatched in FIG. **6**. According to a variant, the rays launched from the second viewpoint **420** have as an end point the second inter-

section with the bounding volume **33**, that is to say, the intersection with the face of the bounding volume farthest from the second viewpoint **420**. For each ray launched and for each voxel belonging to one of these rays, the distance between the origin (that is to say the second viewpoint **420**) and the voxel considered is determined and compared to the distance separating the origin of the intersection point between the surface **32** and the ray considered. If the distance to the voxel is less than the distance to the surface **32**, then the voxel passes to the transparent state. In the contrary case, if the distance to the voxel is greater than the distance to the surface **32**, then the voxel remains in the opaque state. According to this variant the same voxels pass to the transparent state and belong to the areas **433** and **434** illustrated as hatched in FIG. **5**. This second step is used to define the geometrically inconsistent voxels according to the second viewpoint **420**, that is to say the voxels that do not belong to an object of the scene represented by the image **12** associated with the second viewpoint **420**. The voxels whose attribute representative of transparency has passed to the transparent state during the second step are illustrated in white (in the areas **431**, **432**) with respect to FIG. **6**.

FIG. **9** illustrates a fourth step of the method for modelling the scene to generate a surface **90** resulting from the processing performed on the surfaces **31** and **32** of the bounding volume **33** according to the first, second, and third steps of the method illustrated with respect to FIGS. **4**, **5** and **6**. After the first, second and third steps previously described, the attribute representative of transparency associated with each voxel of the bounding volume **33** is either in the transparent state (geometrically inconsistent voxel, that is to say not belonging to an object of the scene to be modelled according to at least one of the capture viewpoints **410**, **420** of images **11**, **12** representative of the scene to be modelled), or in the opaque state (geometrically consistent voxel, that is to say a voxel belonging to an object in the scene to be modelled according to any capture viewpoint **410**, **420** of images **11** and **12** representative of the scene to be modelled). During this fourth step, a surface **90** resulting from the fusion of surfaces **31** and **32** is generated. This surface **90** is advantageously generated according to the attributes representative of transparency associated with the voxels of the bounding volume. The surface **90** is obtained by determining the border between the voxels in the transparent state and the voxels in the opaque state. This surface **90** advantageously corresponds to the exterior envelope of the scene to be modelled from the images **11**, **12** representing it. In other words, the surface **90** corresponds to the set of points or elements visible from any viewpoint and corresponds in this regard to the fusion of the surfaces **31**, **32** representative of points/elements visible from a particular viewpoint (respectively **410**, **420**).

FIG. **7** diagrammatically shows an example of material embodiment of a processing unit **7** adapted to the modelling of a 3D scene from data representative of depth and to the generation of display signals of one or more images, according to an example of a particular and non-restrictive implementation of the invention. Unit **7** corresponds for example to a personal computer PC, a laptop or a games console.

The unit **7** comprises the following elements, connected to each other by an address and data bus **75** that also transports a clock signal:

- a microprocessor **71** (or CPU);
- a graphics card **72** comprising:
  - several Graphics Processing Units **720** (or GPUs),
  - a Graphical Random Access Memory (GRAM) **721**;
- a non-volatile memory of ROM type (Read Only Memory) **76**;
- a Random Access Memory (RAM) **77**;
- one or more I/O (Input/Output) devices **74** such as for example a keyboard, a mouse, a webcam, and
- a power supply **78**.

The device **7** also comprises a display device **73** of display screen type directly connected to the graphics card **72** to display notably the synthesised images calculated and composed in the graphics card, for example live. The use of a dedicated bus to connect the display device **73** to the graphics card **72** offers the advantage of having much greater data transmission bitrates and thus reducing the latency time for the display of images composed by the graphics card. According to a variant, a display apparatus is external to the device **6** and is connected to the device **7** by a cable transmitting the display signals. The device **7**, for example the graphics card **72**, comprises a means for transmission or connector (not shown in FIG. **7**) adapted to transmit a display signal to an external display means such as for example an LCD or plasma screen or a video-projector.

It is noted that the word "register" used in the description of memories **72**, **76** and **77** designates in each of the memories mentioned, both a memory zone of low capacity (some binary data) as well as a memory zone of large capacity (enabling a whole program to be stored or all or part of the data representative of data calculated or to be displayed).

When switched-on, the microprocessor **71** loads and executes the instructions of the program contained in the RAM **77**.

The random access memory **77** notably comprises:

- in a register **770**, the operating program of the microprocessor **71** responsible for switching on the device **7**,
- calibration parameters **771** representative of the intrinsic and extrinsic parameters of the acquisition device(s) used to capture images representative of the scene to be modelled;
- the data **772** representative of images representing the captured scene (e.g. RGB data) and representative of depth associated with the captured images (e.g. corresponding to a depth map or to a disparity map);
- parameters of discretisation **773** used for the bounding volume discretisation (e.g. the number of voxels per volume unit; the number of voxels according to one, two or three dimensions x, y, z; the dimensions in x, y and z of a voxel).

The algorithms implementing the steps of the method specific to the invention and described hereafter are stored in the memory GRAM **721** of the graphics card **72** associated with the unit **7** implementing these steps. When switched on and once the data **772** is loaded into the RAM **77**, the graphics processors **720** of the graphics card **72** load these parameters into the GRAM **721** and execute the instructions of these algorithms in the form of "shader" type microprograms using for example HLSL (High Level Shader Language) or GLSL (OpenGL Shading Language).

The random access memory GRAM **721** notably comprises:

- in a register **7210**, the data representative of images representing the captured scene and the data representative of depth associated with the captured images;
- data **7211** representative of surfaces representative of depth data associated with the images (e.g. the coordinates of points/elements of a surface in world space);
- data **7212** representative of voxels (for example the position of the voxel in the bounding volume, the coordinates of the voxel centre, etc.);
- information **7213** representative of attributes representative of transparency associated with voxels (e.g. in the transparent state or the opaque state), and

data **7214** representative of the surface resulting from the fusion of surfaces representative of the depth data (e.g. coordinates of the points/elements of the resulting surface in world space).

According to a variant, a part of the RAM **77** is assigned by the CPU **71** for storage of the values **7211** to **7214** if the memory storage space available in GRAM **721** is insufficient. This variant however causes greater latency time in the composition of an image comprising a representation of the environment **1** composed from microprograms contained in the GPUs as the data must be transmitted from the graphics card to the random access memory **77** passing via the bus **75** for which the transmission capacities are generally inferior to those available in the graphics card for transmission of data from the GPUs to the GRAM and vice-versa.

According to another variant, the power supply **78** is external to the device **4**.

FIG. **8** illustrates a method for modelling a scene from maps representative of depth associated with images representative of the scene to be modelled, implemented in a processing unit **7**, according to an example of non-restrictive implementation of the invention.

During an initialisation step **80**, the different parameters of the unit **7** are updated. In particular, the data representative of images and associated maps representative of depth are initialised in any way.

Then, during a step **81**, for each map representative of depth associated with an image representative of the scene to be modelled (among the plurality of images representing the scene to be modelled according to different viewpoints), a surface representative of the depth map considered. The representative surface is calculated in world space from the depth information expressed in the image space and stored in the depth map concerned. Conversion of depth data from the image space to world space is obtained from the transformation arrays by taking into account the calibration parameters (intrinsic parameters and/or extrinsic parameters) of the acquisition device(s) used to capture images representative of the scene to be modelled. Each depth map is thus transformed into a surface represented by a set of geometrical coordinates in world space, the geometric coordinates being associated with any point of the surface if the surface is represented by points or geometric coordinates being associated with all elements (for example any mesh of a meshing) of the surface if the surface is represented by elements (the coordinates being associated with the vertices of the meshes of the meshing if the elements correspond to meshes).

Then, during a step **82**, a volume is estimated to include all surfaces representative of depth maps associated with the plurality of images representative of the scene. Once the bounding volume is defined, it is discretised into a plurality of voxels from discretisation parameters predefined or adjusted by a user. The density of voxels per volume unit is advantageously selected according to the quality of the desired final display and according to the number of calculations which can be performed by the processing unit during a given time. The higher the number of voxels per volume unit, the higher the quality of the display (of the scene once modelled), but in exchange the volume of calculations required to obtain the display will be high and require high calculation power. If live constraints exist for the modelling of the scene, the discretisation parameters are advantageously selected to obtain the best compromise between the necessary quantity of calculations and the display quality. If the voxels have dimensions which are too high, quantisation defects appear at the surface resulting from the fusion of the surfaces representative of the depth maps.

Then, during a step **83**, an attribute representative of transparency is associated with each voxel of the discretised bounding volume. The attribute representative of transparency advantageously takes two values, namely a first value corresponding to a transparent state of the associated voxel and a second value corresponding to an opaque state of the associated voxel. Advantageously, the second value of the attribute representative of transparency (corresponding to the opaque state) is associated with each voxel of the bounding volume after the discretisation of the bounding volume, this second value corresponding to the initialisation value associated with voxels during their generation. The value of the attribute associated with the voxels of the bounding volume is modified to change to the first value (corresponding to the transparent state) for voxels of the bounding volume corresponding to at least one of the following cases:

- the voxels of the bounding volume which do not belong to the view cones formed from each viewpoint associated with the surfaces representative of depth maps;
- the voxels of the bounding volume located between the viewpoint associated with a surface and the surface considered, for each surface representative of a depth map;

Finally, during a step **84**, a surface resulting from the fusion of surfaces representative of a depth map is generated according to the attributes representative of transparency associated with the voxels of the bounding volume. The resulting surface advantageously corresponds to the border formed between on the one hand the voxels whose associated transparency attribute takes the first value (voxels in a transparent state), and on the other hand the voxels whose associated transparency attribute takes the second value (voxels in the opaque state).

Naturally, the invention is not limited to the embodiments previously described.

In particular, the invention is not limited to a method of scene modelling but also extends to any device implementing this method and notably all devices comprising at least one GPU. The implementation of calculations necessary for modelling the scene is not restricted to an implementation in shader type microprograms but also extends to an implementation in any program type, for example programs that can be executed by a CPU type microprocessor.

According to an advantageous variant of the invention, depth information (for example a depth map or a disparity map) is transmitted to a remote unit (for example a receiver/decoder (or "set-top box"), a mobile phone, tablet, etc.) by any means of wired or wireless transmission (e.g. using a 3G or 4G type interface, Wifi® and a corresponding transmission channel). The depth information is advantageously obtained from the resulting surface generated from the surfaces representative of depth maps associated with the images representing the modelled scene, by converting the coordinates of the resulting surface in the world space into a representation in the image space. Such a variant presents the advantage of optimising the use of the bandwidth by optimising the quantity of information circulating on the transmission channel. In fact, according to this variant, a single piece of depth information (e.g. a single depth map) is transmitted instead of the plurality of depth maps associated with the images of the input video stream. The depth map generated from the resulting surface comprises depth information necessary for the display of the modelled scene by eliminating the information redundancy contained in the different depth maps associated with the images of the input video stream corresponding to different viewpoints and comprising redundant scene information (the scene being captured from several different viewpoints).

The present invention can be used in video games applications for example, whether by programs that can be executed in a PC, laptop computer or in specialised games consoles producing and displaying images live. The processing unit **7** described with respect to FIG. **7** is advantageously equipped with interaction means such as a keyboard and/or joystick, other command modes such as for example vocal recognition also being possible.

The invention claimed is:

1. A method for modelling a scene from a plurality of maps representative of depth, each map representative of depth being associated with a view of the scene according to a viewpoint, said method comprising:

generating, for each map representative of depth of said plurality, a surface representative of said map representative of depth in world space;

determining a volume bounding the surfaces generated in said world space, said bounding volume being discretized into a plurality of voxels,

associating an attribute representative of transparency with the voxels of the bounding volume;

changing a value of the attribute representative of transparency to a transparent state for the voxels of the bounding volume not belonging to view cones formed from each viewpoint associated with said surfaces;

associating an attribute representative of transparency with the voxels of the bounding volume;

generating, for each surface, a plurality of rays originating from a viewpoint associated with said each surface and having as end point an intersection with said each surface;

running through said plurality of generated rays, and changing the value of the attribute representative of transparency for the voxels of the bounding volume having an intersection with one of the generated rays; and

generating data representative of a resulting surface according to the attributes representative of transparency associated with said voxels.

2. The method according to claim **1**, further comprising, for each surface, associating voxels of the bounding volume with said each surface to define said each surface, according to a criterion of proximity of the voxels relative to said each surface, the end points of the generated rays corresponding to the voxels associated with said each surface.

3. The method according to claim **2**, wherein said attribute representative of transparency comprises two states, a first state corresponding to the transparent state of the voxel associated with said attribute, a second state corresponding to an opaque state of the voxel associated with said attribute, a voxel being in the transparent state when said voxel has an intersection with a ray of said plurality of rays.

4. The method according to claim **1**, wherein said attribute representative of transparency is determined according to a position of the voxel associated with said attribute with respect to a viewpoint and to the surface representative of a depth map associated with the viewpoint.

5. The method according to claim **4**, wherein said attribute comprises two states, a first state corresponding to the transparent state of the voxel associated with said attribute, a second state corresponding to an opaque state of the voxel associated with said attribute, a voxel being in the transparent state when positioned between a viewpoint and the surface representative of the depth map associated with the viewpoint in said world space.

6. The method according to claim **4**, wherein the position of the voxel is obtained by determining an intersection between said voxel and a ray launched from said viewpoint.

7. The method according to claim **1**, wherein said surface representative of the depth map in the world space is generated using parameters representative of calibration associated with at least one camera used to capture the views of the scene associated with the plurality of depth maps.

8. The method according to claim **1**, wherein the plurality of depth maps is obtained from a multi-view video stream.

9. The method according to claim **8**, wherein the video stream is an MVD (Multiview Video plus Depth) stream.

10. The method according to claim **1**, further comprising transmitting depth information representative of said resulting surface.

11. A device for processing data representative of a plurality of maps representative of depth, each map representative of depth being associated with a view of a scene according to a viewpoint, wherein the device comprises

at least one processor configured for:

generating, for each map representative of depth of said plurality, a surface representative of said map representative of depth in world space;

determining a volume bounding all surfaces generated in said world space, the said bounding volume being discretized into a plurality of voxels,

associating an attribute representative of transparency being associated with the voxels of the bounding volume;

changing a value of the attribute representative of transparency to a transparent state for the voxels of the bounding volume not belonging to view cones formed from each viewpoint associated with said surfaces;

generating, for each surface, a plurality of rays originating from the viewpoint associated with said each surface and having as an end point the intersection with said each surface;

running through said plurality of generated rays, and changing the value of the attribute representative of transparency for the voxels of the bounding volume having an intersection with one of the generated rays; and

generating data representative of a resulting surface according to the attributes representative of transparency associated with said voxels.

12. The device according to claim **11**, wherein the at least one processor is a Graphics Processing Unit.

13. The device according to claim **11**, wherein the at least one processor is further configured to, for each surface, associate voxels of the bounding volume with said generated surface to define said each surface, according to a criterion of proximity of the voxels relative to said each surface, the end points of the generated rays corresponding to the voxels associated with said each surface.

14. The device according to claim **11**, wherein said attribute representative of transparency comprises two states, a first state corresponding to the transparent state of the voxel associated with said attribute, a second state corresponding to an opaque state of the voxel associated with said attribute, a voxel being in the transparent state when said voxel has an intersection with a ray of said plurality of rays.

15. The device according to claim **11**, wherein said attribute representative of transparency is determined according to a position of the voxel associated with said attribute with respect to a viewpoint and to the surface representative of a depth map associated with the viewpoint.

16. The device according to claim 15, wherein said attribute comprises two states, a first state corresponding to the transparent state of the voxel associated with said attribute, a second state corresponding to an opaque state of the voxel associated with said attribute, a voxel being in the transparent state when positioned between a viewpoint and the surface representative of the depth map associated with the viewpoint in said world space.

17. The device according to claim 15, wherein the position of the voxel is obtained by determining an intersection between said voxel and a ray launched from said viewpoint.

18. The device according to claim 11, wherein said surface representative of the depth map in the world space is generated using parameters representative of calibration associated with at least one camera used to capture the views of the scene associated with the plurality of depth maps.

19. The device according to claim 11, wherein the plurality of depth maps is obtained from a multi-view video stream.

20. The device according to claim 19, wherein the video stream is an MVD (Multiview Video plus Depth) stream.

21. The device according to claim 11, wherein the at least one processor is further configured to transmit depth information representative of said resulting surface.

22. A method for modeling a scene from a plurality of maps representative of depth, each map representative of depth being associated with a view of the scene according to a viewpoint, said method comprising:

generating, for each map representative of depth of said plurality, a surface representative of said map representative of depth in world space;

determining a volume bounding surfaces generated in said world space, said bounding volume being discretized into a plurality of voxels,

associating an attribute representative of transparency with the voxels of the bounding volume;

changing a value of the attribute representative of transparency to a transparent state for the voxels of the bounding volume not belonging to view cones formed from each viewpoint associated with said surfaces;

generating, for each surface, a plurality of rays originating from the viewpoint associated with said each surface and having as end point an intersection with said each surface;

running through said plurality of generated rays, and changing the value of the attribute representative of transparency for the voxels of the bounding volume having an intersection with one of the generated rays;

generating data representative of a resulting surface according to the attributes representative of transparency associated with said voxels; and

displaying the modeled scene on a display device.

23. A device for processing data representative of a plurality of maps representative of depth, each map representative of depth being associated with a view of a scene according to a viewpoint, wherein the device comprises

at least one processor configured for:

generating, for each map representative of depth of said plurality, a surface representative of said map representative of depth in world space;

determining a volume bounding all surfaces generated in said world space, the said bounding volume being discretized into a plurality of voxels, an attribute representative of transparency being associated with the voxels of the bounding volume;

changing a value of the attribute representative of transparency to a transparent state for the voxels of the bounding volume not belonging to view cones formed from each viewpoint associated with said surfaces;

generating, for each surface, a plurality of rays originating from a viewpoint associated with said each surface and having as an end point the intersection with said each surface;

running through said plurality of generated rays, and changing the value of the attribute representative of transparency for the voxels of the bounding volume having an intersection with one of the generated rays;

generating data representative of a resulting surface according to the attributes representative of transparency associated with said voxels; and

displaying the scene on a display device.

* * * * *

US 20140354632A1

(19) **United States**

(12) **Patent Application Publication**

(10) **Pub. No.: US 2014/0354632 A1**

(43) **Pub. Date: Dec. 4, 2014**

Alj et al.

(54) **METHOD FOR MULTI-VIEW MESH TEXTURING AND CORRESPONDING DEVICE**

(71) Applicant: **THOMSON LICENSING**, Issy de Moulineaux (FR)

(72) Inventors: **Youssef Alj**, Rennes (FR); **Guillaum Boisson**, Pleumeleuc (FR); **Philippe Bordes**, Laille (FR); **Luce Morin**, Rennes (FR); **Muriel Pressigout**, Rennes (FR)

(73) Assignee: **Thomson Licensing**, Issy de Moulineaux (FR)

(21) Appl. No.: **14/372,018**

(22) PCT Filed: **Oct. 8, 2012**

(86) PCT No.: **PCT/EP2012/069862**

§ 371 (c)(1),
(2), (4) Date: **Jul. 13, 2014**

(30) **Foreign Application Priority Data**

Jan. 13, 2012 (EP) 12305047.8
Oct. 8, 2012 (WO) PCTEP2012/069862

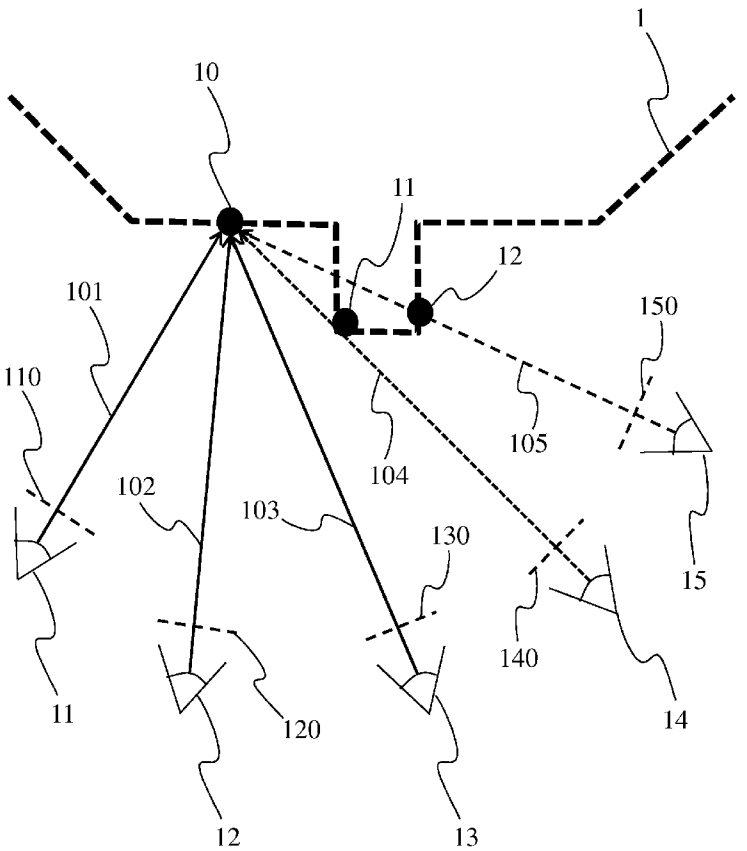
Publication Classification

(51) **Int. Cl.**
G06T 15/04 (2006.01)
H04N 13/02 (2006.01)

(52) **U.S. Cl.**
CPC ***G06T 15/04*** (2013.01); ***H04N 13/0282*** (2013.01); ***H04N 13/0275*** (2013.01)
USPC **345/419**

(57) **ABSTRACT**

A method for texturing a mesh associated with a surface representative of a scene captured in a plurality of images, wherein at least a mesh element of the mesh is at least partially visible from at least two first images of the plurality of images. As to reduce the amount of texture information associated with multi-views data representative of the scene, the method comprises for each first image (**110** to **140**), association of a first texture information with the mesh element, projection of the first texture information in the at least two first images, for each first texture information, estimation of an error information according to a comparison result between the projected first texture information and the texture information of said at least two first images, and selection of one of the at least two first texture information according to the error information.



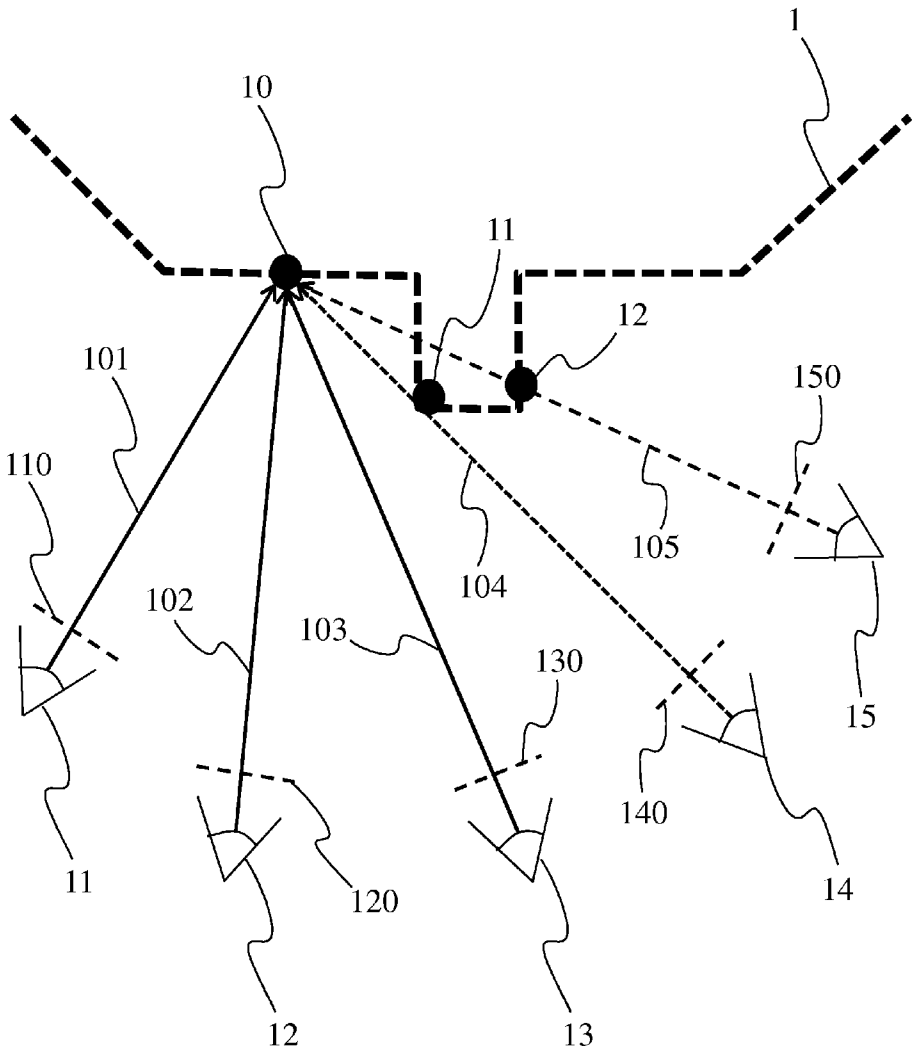


Fig 1

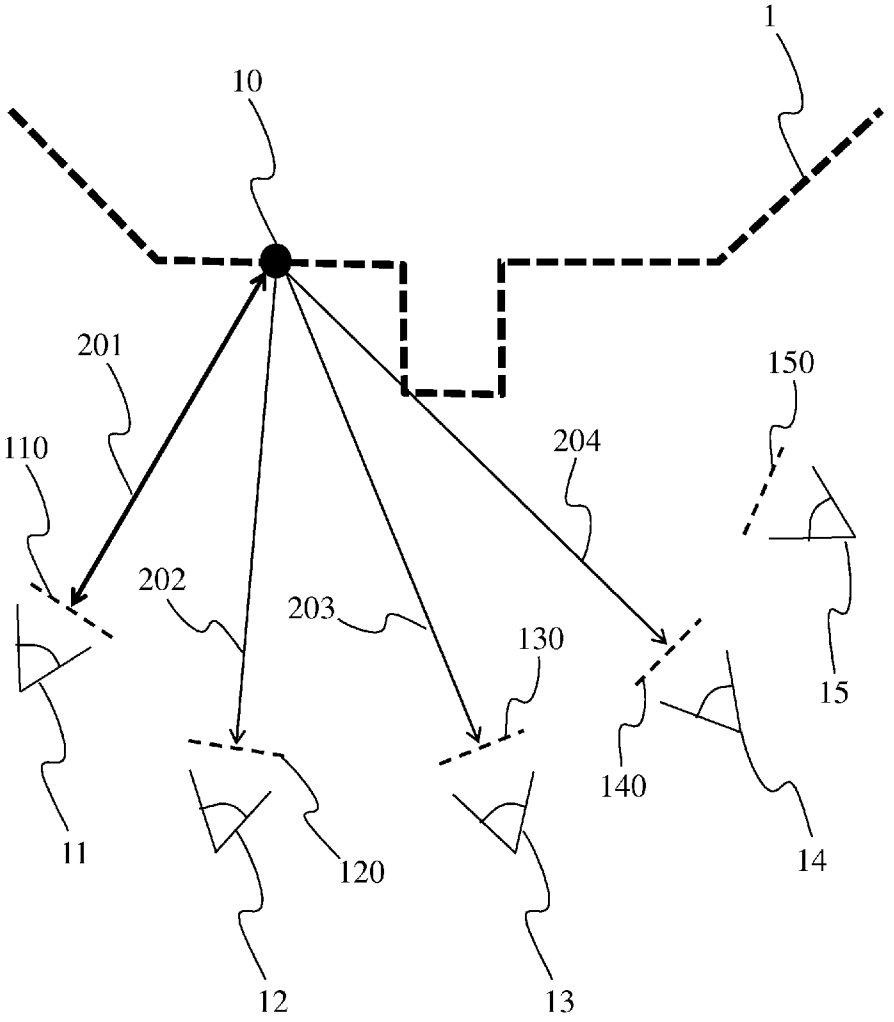


Fig 2

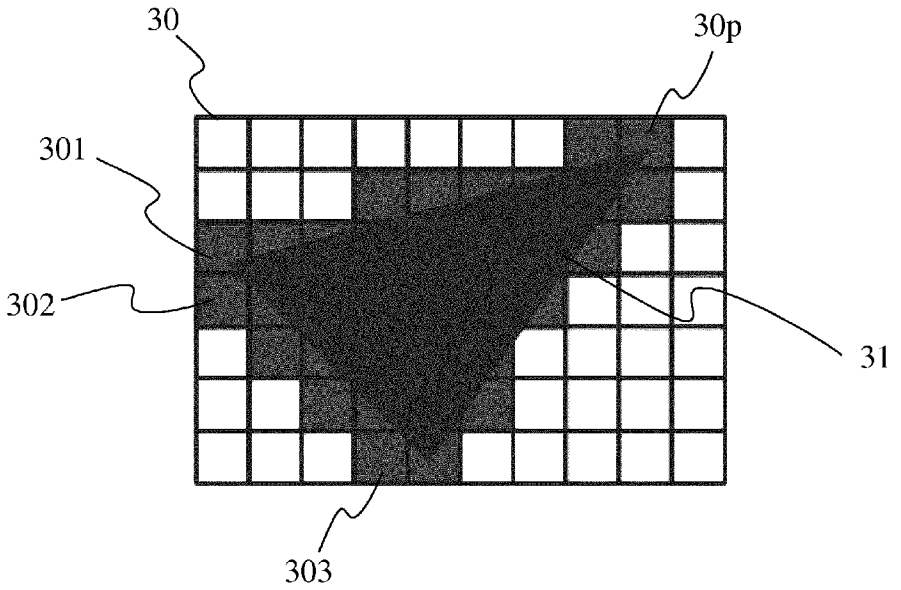


Fig 3

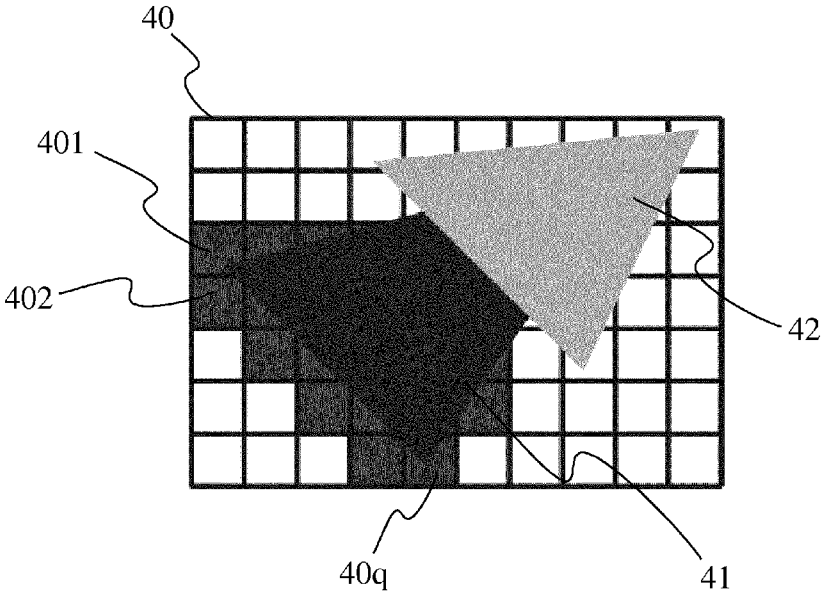


Fig 4

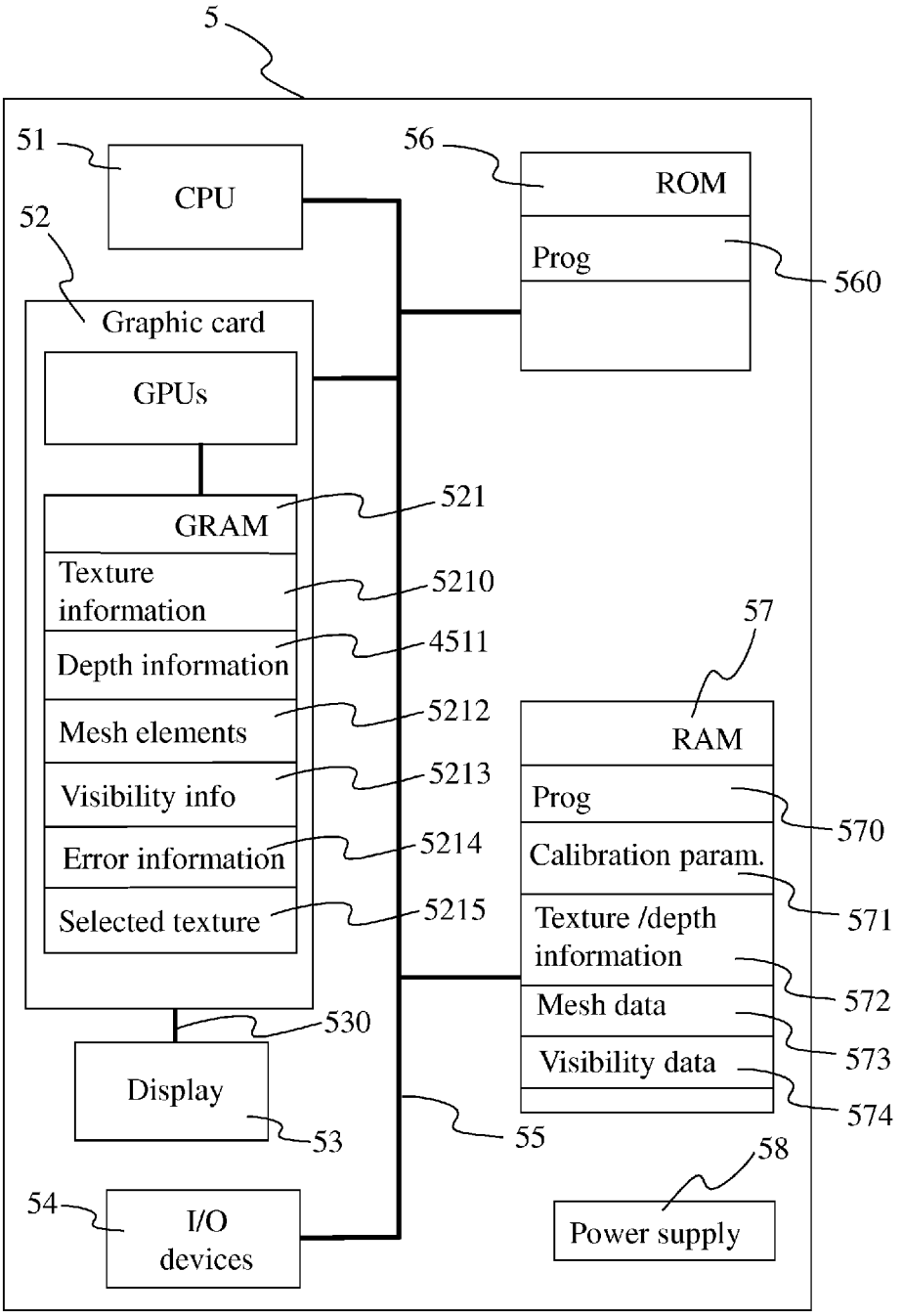

5
51
CPU
56
ROM
Prog
560
52
Graphic card
GPUs
GRAM
521
Texture information
5210
Depth information
4511
Mesh elements
5212
Visibility info
5213
Error information
5214
Selected texture
5215
57
RAM
Prog
570
Calibration param.
571
Texture /depth information
572
Mesh data
573
Visibility data
574
530
Display
53
55
58
Power supply
54
I/O devices

Fig 5

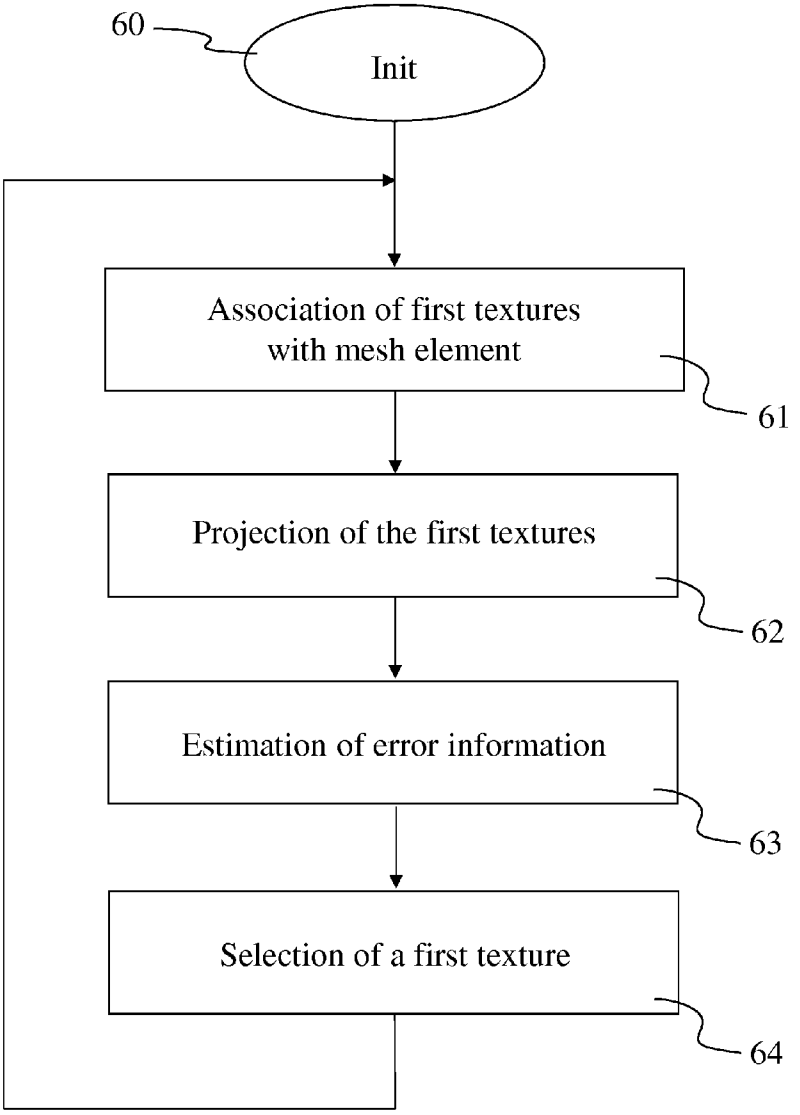

60
Init
Association of first textures with mesh element
61
Projection of the first textures
62
Estimation of error information
63
Selection of a first texture
64

Fig 6

METHOD FOR MULTI-VIEW MESH TEXTURING AND CORRESPONDING DEVICE

1. SCOPE OF THE INVENTION

[0001] The invention relates to the domain of image or video processing and more particularly in the processing of data representative of texture associated to images of a multi-views video stream. The invention also relates to the domain of modelling the texture of real scene by using texture data and depth data associated to images representative of the scene according to several points of view.

2. PRIOR ART

[0002] Combined with the use of stereoscopic displays, Multi-View Imaging (MVI) provides a realistic depth perception to the user and allows a virtual navigation around the scene. It also offers the possibility to synthesize virtual views, opening therefore a broad spectrum of applications such as 3DTV and Free-viewpoint TV (FTV). An end-to-end MVI chain presents many challenges due to the inherent problems related to scene capture, data representation and transmission. Indeed, during scene acquisition, multiple cameras are used; hence several photometric errors may occur due to the changing illumination across different views, which increase the signal ambiguity during depth estimation. Scene representation is also a challenging task. One should strive to build a dense yet non-redundant scene model from the set of overlapping views, each view comprising information related to the texture and to the depth. The compromise between model simplicity and easy rendering is usually hard to satisfy. The representation is then to be transmitted through a communication channel. Determining the proper rate-distortion trade-off is another key point. At decoder side, high fidelity to original views is required. Finally to assess the quality of virtual views, conventional image based objective metrics such as Peak of Signal to Noise Ratio (PSNR) or Structural Similarity (SSIM) are not applicable since the reference images for such views are not available.

3. SUMMARY OF THE INVENTION

[0003] The purpose of the invention is to overcome at least one of these disadvantages of the prior art.

[0004] More particularly, the invention has the notable purpose of reducing the amount of texture information associated to multi-views data representative of a scene.

[0005] The invention relates to a method for texturing a mesh, the mesh being associated with a surface representative of a scene captured according to a plurality of points of view, an image of the scene comprising texture information being associated with each point of view, wherein at least a mesh element of the mesh is at least partially visible from at least two first images of the plurality of images. The method comprises the following steps:

[0006] for each first image, association of a first texture information with the mesh element,

[0007] projection of the first texture information in the at least two first images,

[0008] for each first texture information, estimation of an error information according to a comparison result between the projected first texture information and the texture information of the at least two first images, and

[0009] selection of one of the at least two first texture information according to said error information.

[0010] Advantageously, the error information associated to a first texture information corresponds to a sum of the comparison results between the projected first texture information and each of the at least two first images.

[0011] According to a particular characteristic, the selected first texture information corresponds to the first texture information having the least error information.

[0012] Advantageously, the method further comprises the steps of:

[0013] projection of the mesh element in at least a part of the images of the scene,

[0014] for each image of the at least a part of images, comparison between a depth information associated with the projected mesh element and a depth information associated with the image,

[0015] for each image of said at least a part of images, determination of a visibility information associated to said mesh element (t) according to the result of the comparison of previous step, a mesh element being labeled as at least partially visible or not visible from said image according to said visibility information.

[0016] Advantageously, the projection of the mesh element corresponds to a conversion of the coordinates of the mesh element from world space into the image space of the image.

[0017] According to another characteristic, each image comprises a plurality of pixels, video information being associated to the plurality of pixels.

[0018] According to a specific characteristic, each image is represented by using Multi-view Video plus Depth (MVD) representation.

[0019] The invention also relates to a computation unit configured to texture a mesh, the mesh being associated with a surface (**1**) representative of a scene captured according to a plurality of images, each image comprising texture information, wherein at least a mesh element of the mesh is at least partially visible from at least two first images of the plurality of images, the computation unit comprising:

[0020] means for associating, for each first image, a first texture information with the mesh element,

[0021] means for projecting the first texture information in the at least two first images,

[0022] means for estimating, for each first texture information, an error information according to a comparison result between the projected first texture information and the texture information of the at least two first images, and

[0023] means for selecting one of the at least two first texture information according to the error information.

[0024] Advantageously, the error information associated to a first texture information corresponds to a sum of the comparison results between the projected first texture information and each of the at least two first images.

[0025] According to another characteristic, the selected first texture information corresponds to the first texture information having the least error information.

[0026] According to a particular characteristic, the computation unit further comprises:

[0027] means for projecting the mesh element in at least a part of the images of the scene,

[0028] means for comparing, for each image of the at least a part of images, between a depth information

associated with the projected mesh element and a depth information associated with the image,

[0029] means for determining, for each image at said least a part of the images, a visibility information associated to the mesh element according to the result of the comparison, a mesh element being labeled as at least partially visible or not visible in the image according to the visibility information.

[0030] According to a specific characteristic, the means for projecting said mesh element comprise means for converting the coordinates of the mesh element from world space into the image space of the image.

4. LIST OF FIGURES

[0031] The invention will be better understood, and other specific features and advantages will emerge upon reading the following description, the description making reference to the annexed drawings wherein:

[0032] FIG. **1** illustrates a meshed surface **1** representative of a scene viewed and captured according to several points of views, according to a particular embodiment of the invention;

[0033] FIG. **2** illustrates the association of a first texture information with a mesh element of the surface **1** of FIG. **1** and the projection of the first texture information on at least a part of the views, according to a particular embodiment of the invention;

[0034] FIG. **3** illustrates a mesh element of the meshed surface **1** of FIG. **1** entirely visible according to at least one of the points of view, according to a particular embodiment of the invention;

[0035] FIG. **4** illustrates a mesh element of the meshed surface of FIG. **1** partially visible according to at least one of the points of view, according to a particular embodiment of the invention;

[0036] FIG. **5** illustrates a device implementing a method for texturing a mesh of surface **1** of FIG. **1**, according to a particular implementation of the invention;

[0037] FIG. **6** illustrates a method for texturing a mesh of a surface **1** of FIG. **1**, according to a particular implementation of the invention.

5. DETAILED DESCRIPTION OF EMBODIMENTS OF THE INVENTION

[0038] The invention will be described in reference to a particular embodiment of a method for texturing a mesh of surface, which is representative of the scene. The scene has been acquired from several points of view, an image or view of the scene being associated with a point of view. A mesh is associated with the surface, the mesh comprising a plurality of mesh elements (for example triangles or quadrilaterals). Each image of the scene comprises a plurality of pixels, the number of pixels depending on the resolution of the image of the scene associated with the view. A texture information is associated with each image of the scene, the texture information corresponding for example to a video information (for example RGB data representative of the grey level associated with each colour (RGB) information) associated with the pixels of the image corresponding to a view of the scene. For a mesh element which is at least partially visible from two or more views or, equivalently, in two or more images (called first images in the following description), a first texture information extracted from one of the first images is associated to the mesh element and projected on all or part of the first image, i.e. on the image where the mesh element is at least partially visible. The projected first texture information is compared with each texture information associated to the first images on which the first texture information has been projected. From the comparison results, an error information is estimated which is for example representative of the difference between the projected first texture information and the texture information associated with the first images on which the first texture information has been projected. The step of associating a first texture information with the mesh element is reproduced for another first image and the steps of projecting and comparing this new first texture information are reproduced. These steps are advantageously reiterated for each first image, forming a set of several candidate first texture information. When an error information has been estimated for all (candidate) first texture information, one of them is selected according to the error information assigned with each of them. The selection of one single texture information (to be assigned with a mesh element) among the plurality of texture information associated with the plurality of views for which the mesh element is at least partially visible enables to reduce the amount of data to be transmitted to a display or any receiver over a network. According to the invention, minimal input data needed to associate a texture with a mesh element is a meshed surface representative of a scene and texture information associated with a plurality of images representing the scenes according to different points of view. It also enables to select the best texture information candidate to map a given mesh element, reducing photometric errors, which occur for example when the texture information associated with a mesh element correspond to the average of all texture information available in the different views where the mesh element is visible or partially visible.

[0039] FIG. **1** illustrates a surface **1** representative of a scene viewed and captured according to several points of views. The surface **1** is representative of a scene acquired according to a plurality of points of view **11**, **12**, **13**, **14**, **15**. An image **110**, **120**, **130**, **140**, **150** is associated with each point of view **11** to **15**, corresponding to a representation of the scene according to a given point of view. A mesh is associated with the surface **1**, the mesh comprising a plurality of mesh elements (for example thousands of mesh elements), for example triangles or quadrilaterals. The surface **1** corresponds advantageously to a 3D representation of the scene.

[0040] The surface **1** and the associated mesh are obtained by combining methods well-known by the skilled person in the art, for example by combining the "shape from silhouette" method with the marching-cubes algorithm or by combining the space carving method with the marching-cubes algorithm. The "shape from silhouette" method (described for example in "*Spatio-Temporal Shape from Silhouette using Four-Dimensional Delaunay Meshing*" by Ehsan Aganj, Jean-Philippe Pons, Florent Ségonne and Renaud Keriven) and the space carving method (described in "*A Theory of Shape by Space Carving*" published in *International Journal of Computer Vision,* 38(3), 199-219 (2000), by Kiriakos N. Kutulakos and Steven M. Seitz) are used for obtaining a 3D representation of a scene from a plurality of views of the scene. The marching cubes algorithm (published in the 1987 *SIGGRAPH proceedings* by Lorensen and Cline) is then applied to the 3D representation of the scene as to obtain a meshed surface as a 3D representation of the scene. French Patent application FR1161788, filed on Dec. 16, 2011, describes a method for obtaining a meshed surface corre-

sponding to a 3D representation of a scene. FR1161788 recites a 3D modeling system designed for Multi-view Video plus Depth (MVD) sequences. The aim is to remove redundancy in depth information present in the MVD data. To this end, a volumetric framework is employed in order to merge the input depth maps. Hereby a variant of the space carving algorithm is proposed: voxels are iteratively carved by ray-casting from each view, until the 3D model be geometrically consistent with every input depth map. A surface mesh is then extracted from this volumetric representation thanks to the marching cubes algorithm.

[0041] At least one of the mesh elements is visible or partially visible in different images associated to the views of the scene. According to the non limitative example of FIG. **1**, the mesh element **10** is entirely visible from the points of view **11**, **12**, **13**, i.e. the mesh element **10** is entirely visible in the first images **110**, **120**, **130** associated with the points of view **11**, **12**, **13**. The mesh element **10** is partially visible from the point of view **14**, i.e. the mesh element **10** is partially visible in the first image **140**. Indeed, as illustrated on FIG. **1**, the viewing direction **104** having as origin the point of view **14** and as extremity the mesh element **10** is tangent to a point **11** (or mesh element **11**) of the surface **1** on the path separating the point of view **14** and the mesh element **10**. The mesh element is not visible from the point of view **15** and is not visible on the associated image **150**. The image **150** is thus not considered as being a first image. Indeed, as illustrated on FIG. **1**, the viewing direction **105** having as origin the point of view **15** and as extremity the mesh element **10** has an intersection with the surface **1**, the intersection corresponding to the point or mesh element **12** of surface **1**, the intersection **12** being positioned between the point of view **15** and the mesh element **10** on the path corresponding to the viewing direction **105**. For each image **110** to **150**, a visibility label is advantageously associated with the mesh element **10**. The visibility label takes for example two values, i.e. visible and non visible, or three value, i.e. entirely visible, partially visible and non visible.

[0042] According to a non limitative embodiment, the visibility information associated to the mesh element **10** is determined by comparing a depth information associated with the mesh element **10** and a depth information associated to pixels of a first image **110** to **150** on which the mesh element **10** is projected. The mesh element visibility with respect to each image **110** to **150** is determined for example using OpenGL z-buffer. In a first pass, the mesh (i.e. all mesh elements) is projected onto a current image (for example the image **110**), and the z-buffer is extracted, the z-buffer comprising a depth information associated to each mesh element as seen in the current image (for example, a depth information is associated with each pixel of the current image). Depth information is determined by projecting mesh elements into an image, i.e. by converting the coordinates of the mesh element from world space into the image space. A second pass is dedicated to visibility determination using the computed z-buffer. Each vertex of the mesh element **10** is projected onto the current image **110** and the depth component, denoted as $z_{projected}$, is checked against the pixel depth z-buffer of the current image **110**. If the projected vertex is behind the pixel in the z-buffer (i.e. the value $z_{projected}$ is greater than the depth value associated to the pixel in the z-buffer of the current image), then this vertex is hidden, and thus the set of mesh triangles lying on this vertex are not visible. In contrast, if the projected vertex is ahead the pixel in the z-buffer (i.e. the value $z_{projected}$ is less than the depth value associated to the pixel in the z-buffer of the current image), then this vertex is visible, and thus the set of mesh triangles lying on this vertex are at least partially visible. The visibility information associated with the mesh element is for example determined according to the following algorithm:

```
// Views traversal
for each view V_j do
    Initialize all mesh elements to visible.
    Project M onto V_j .
    Read z_buffer.
    // Mesh vertices traversal
    for each vertex v do
        Determine (q, l, z_projected) the projection of the vertex v(X, Y,Z)
        onto V_j .
        if z_buffer[q,l] > z_projected then
            v is a hidden vertex. Mark all the triangles lying on v as
            hidden.
        end
    end
end
```

[0043] According to a variant, the mesh is not projected into the current image as to determine the z-buffer associated with the current image. The depth of the mesh element is compared to a depth information comprised in a depth map associated with the current image, the depth map being received with the image (for example in a stream of the type MVD).

[0044] According to a variant, the visibility information associated with the mesh element **10** (and with each mesh element of the surface **1**) is received with the data representatives of the images **110** to **150** and is not to be determined. These data comprises for example RGB information for each pixel and visibility information associated with the mesh elements of the mesh of FIG. **1** and comprising information about the visibility of mesh elements in each and every images representing the scene (the visibility information being for example stored in a visibility map).

[0045] Images **110** to **150** belong to a multi-view video stream comprising a sequence of several images representative of a same scene. According to a non-limitative example, the multi-view video stream is MVD type (Multi-view Video plus Depth). In case of MVD data, depth information is associated with each image representing the scene. A depth information is thus associated with each image **110** to **150**, the depth information correspond to data representative of depth, for example a depth map (for example of the type of z-buffer) or a disparity map. A depth information is a generic name and corresponds to any data structure representative of a depth or of a disparity. The depth information associated with the pixels of the images **110** to **150** are for example captured with the use of appropriate sensors, for example by using infrared sensors associated with an infrared emitter (for example system of the type Kinect®) or by using a system comprising a laser emitter and an associated sensor configured for determining the crossing time of a laser ray emitted toward the captured scene and reflected by the scene, the determined time being representative of the path crossed and thus of the depth associated with the point of the scene having reflected the laser ray. According to a variant, video information (corresponding for example to colour information) and depth information associated with the pixels of images **110** to **150** are stored in memories under the form of RGBα (Red, Green, Blue, α channels) information, channels RGB being used for

storing video information associated with each pixel (for example 8, 10 or 12 bits per channel) and the α channel being used for storing the depth information (or disparity information), for example on 8, 10 or 12 bits.

[0046] Advantageously, intrinsic and extrinsic parameters of acquisition devices used for acquiring the images **110** to **150** are known and for example stored in a memory. Intrinsic parameters comprise for example focal length, enlargement factor of the image, coordinates of the projection of the optical centre of the acquisition device on image plane and/or a parameter representative of potential non-orthogonality of lines and columns of photosensitive cells forming the acquisition device. Extrinsic parameters comprise for example the orientation matrix for passing from world space to image space (and inversely) and/or components of translation vector for passing from world space to image space (and inversely).

[0047] Naturally, the number of images is not limited to 5 images but extends to any number greater than 2, for example 2, 3, 4, 5, 10, 20, 50, 100 images, data representative of depth being associated with each image. In a same way, the number of first images is not limited to 4 but extends to any number greater than 2. Advantageously, each image is acquired by a particular acquisition device according to a particular point of view, acquisition devices being spatially staggered. According to a variant, only a pair of left and right images is acquired by means of two acquisition devices, the other images of the plurality of images representative of the acquired scene being estimated from the left and right images by disparity compensated interpolation. According to this variant, each image is also representative of the same scene according to a particular point of view and depth information is associated with each image, independently of the fact that the image be acquired or interpolated.

[0048] FIG. **2** illustrates the association of a first texture information with a mesh element of the surface **1** and the projection of a first texture information on some views, according to a particular and non limitative embodiment of the invention. The mesh element **10** is visible or partially visible from the points of view **11**, **12**, **13** and **14**, i.e. the mesh element **10** is visible or partially visible in the images **110**, **120**, **130** and **140** respectively associated with the points of view **11**, **12**, **13** and **14**. The images **110** to **140** in which the mesh element is partially visible are called first images. One first image **110** is selected among the plurality of first images **110** to **140** and a first texture information extracted from thus first image **110** is extracted and associated with the mesh elements. The first texture information corresponds advantageously to the video information (or RGB colour information) associated with the pixels of the first image **110** corresponding to the projection of the mesh element **10** onto the first image **110**. The pixels of the first image on which is projected the mesh element **10** are illustrated on FIG. **3**. According to a variant, the texture information associated with the images corresponds to YUV data.

[0049] FIG. **3** illustrates the projection **31** of the mesh element **10** onto an image **30** (corresponding for example to the image **110**) on which the mesh element is entirely visible. The first texture information associated with the mesh element **10** corresponds to the video information associated with the pixels on which the mesh element **10** is projected, these pixels being "covered" or partially "covered" by the projection **31** of the mesh elements. These pixels **301** to **30***i* are illustrated in grey on FIG. **3**.

[0050] As illustrated on FIG. **2** by the double arrow **201**, the mesh element **10** is first textured with a first texture information of image **110**. This first texture information is then projected onto all first images **110** to **140**, which is illustrated by the arrows **201**, **202**, **203** and **204**. The projection of the mesh element **10** (textured with the first texture information of corresponding pixels of first image **110**) onto first images **110**, **120**, **130** for which the mesh element **10** is entirely visible is illustrated by FIG. **3** and the projection of the mesh element **10** (textured with the first texture information of corresponding pixels of first image **110**) onto the first image **140** for which the mesh element **10** is only partially visible is illustrated by FIG. **4**. The projection of the mesh element **10** onto one first image **110** to **140** corresponds to a conversion of the coordinates of the mesh element **10** from the world space into the image space associated to the first image onto which the mesh element is projected.

[0051] On FIG. **3**, the image **30** corresponds to a first image (first images **110**, **120** or **130**) on which the mesh element **10** is projected. The projection of the mesh element **10** corresponds to the mesh element referenced **31**. Pixels in gray **301** to **30***p* represents the p pixels (p being an integer) of the projection image covered at least partially by the projection **31** of the mesh element **10**. On FIG. **4**, the image **40** corresponds to the first image **140** of FIG. **2** on which the mesh element is only partially visible, a part of the mesh element **10** being hidden by another mesh element **11** illustrated on FIG. **1**. The projection of the mesh element **10** on image **40** is referenced **41** and the projection of the mesh element **11** on image **40** is referenced **42**. According to this example, the number q (q being an integer) of pixels **401** to **40***q* of image **40** at least partially covered by the projection **41** of the mesh element **10** are less than the number p of the pixels of image **30** (p>q) (with a same resolution for all first images **110** to **140**).

[0052] After the projection of the first texture information of first image **110** associated with the mesh element **10** on the first images **110** to **140**, the first texture information is compared with the texture information associated with the pixels of each first image **110** to **140** corresponding to the projection of the mesh elements. The pixels of a first image **110** to **140** corresponding to the projection of the mesh element **10** also correspond to the pixels of this first image for which the mesh element is visible according to the point of view associated with this image. The result of the comparison advantageously corresponds to the distortion between the projected first texture information and the texture information of the pixels of the image onto which the mesh element is projected, and is:

$$D_{i,j}=\|M_{I_i\rightarrow V_j}-I_j\|^2 \qquad \text{equation 1}$$

[0053] Where:

[0054] $M_{I_i\rightarrow V_j}$ corresponds to the projection onto the image V_j (for example one of the image **120**, **130** or **140**) of the mesh element **10** textured with the first texture information of image I_i (for example first image **110**)

[0055] $D_{i,j}$ corresponds to the distortion between the texture information of images i and j.

[0056] The first texture information of image **110** associated with the mesh element **10** is compared with each texture information of each first image **110** to **140**. According to a variant, the first texture information is compared with each texture information of each first image except with the first image **110**, the distortion between the projected first texture

information (of first image **110**) and the texture information of image **110** being equal to 0 or close to 0.

[0057] The same process is repeated by associating a first texture information extracted from the first image **120** with the mesh element **10**, by projecting this new first texture information onto the first images **110** to **140** and by comparing it with the texture information associated with the pixels corresponding to the projection of the mesh element **10** onto the first images **110** to **140** as to determine the distortion between the projected first texture information and each first image. This process is reiterated by associating first texture information extracted from each first image **130** and **140** as to determine the distortion between each projected first texture information and each first image. According to a variant, the distortion is determined between each first texture information (from each first image **110** to **140**) with only a part of the first images (by excepting the first image providing the first texture information to be projected onto the first images for comparison). A non limitative example of an algorithm used for determining the result of comparison between the first texture information and the first images is as follow:

```
// Textures traversal
for each texture I_i do
    // Views traversal
    for each view V_j do
        Compute M_{I_i→V_j}: projection of the mesh textured with texture I_i
        onto view V_j.
        D_{i,j} = ||M_{I_i→V_j} − I_j||^2
    end
end
```

[0058] For each first texture information associated with the mesh element **10** (extracted from each first image **110** to **140**), an error is determined and associated with respectively each first texture information. The determination of the error associated with a first texture information us based on the comparison results between the projected first texture information and the texture information associated with the pixels of the first images onto which the textured mesh element is projected. The calculation of an error information associated with the mesh element (T) 10 textured with a first texture information originating from a first image I_i (i corresponding to a first image, i being equal to 4 according to the non limitative example of FIGS. **1** and **2**) is performed for example by using the following equation:

$$Error_{T,I_i} = \sum_{\substack{j=1 \\ T\ visible\ in\ V_j}}^{n} \left(\sum_{(q,l)\in M_{T,V_j}} D_{i,j}[u, v] \right) \qquad \text{equation 2}$$

[0059] u, v corresponding to the coordinates of pixel in an image I having a resolution of q×I pixels.

[0060] One first texture information is then selected among the plurality of first texture information according to the errors associated with each first texture information. The selected first texture information corresponds advantageously to the first texture information having the least error value. A first texture information is for example selected according to the following equation:

$$\hat{I}_T = \text{Error}_{T,I_i}(I_i \in \mathcal{F})$$

[0061] $\mathcal{F}$ corresponding to the set of first texture information (available in the first images **110** to **140**).

[0062] A non limitative example of an algorithm used for selecting the first texture information which minimizes the error is as follow:

```
// Triangles traversal
for each triangle T do
    // Views traversal
    for each view V_j do
        Determine M_{I_i→V_j} the projection of the triangle T onto V_j where
        T is visible.
    end
    // Textures traversal
    for each texture I_i do
        Compute Error_{T,Ii}.
    end
        Determine Î_T.
end
```

[0063] The selection of a first texture information among the set of candidate first texture information enables to assign one single texture information to a mesh element for every image in which the mesh element is partially visible, which reduces the amount of data to be coded and transmitted to any display device displaying the images or to any device decoding the data representative of the images to be displayed. The selection of the first texture information minimizing photometric error also enables to reduce artefacts which could appear if the texture associated to a mesh element would correspond to an arbitrary selection of one candidate texture information or to a mix of the candidate texture information.

[0064] Advantageously, a first texture information is assigned to each mesh element of the mesh or to a part of the mesh elements of the mesh.

[0065] FIG. **5** diagrammatically shows a hardware embodiment of a device **5** adapted and configured for texturing a mesh of a surface **1** and for the generation of display signals of one or several images, according to a particular and non limitative embodiment of the invention. The device **5** corresponds for example to a personal computer PC, a laptop, a tablet or a mobile phone.

[0066] The device **5** comprises the following elements, connected to each other by a bus **55** of addresses and data that also transports a clock signal:

- **[0067]** a microprocessor **51** (or CPU),
- **[0068]** a graphics card **52** comprising:
  - **[0069]** several Graphical Processor Units (or GPUs) **520**,
  - **[0070]** a Graphical Random Access Memory (GRAM) **521**,
- **[0071]** a non-volatile memory of ROM (Read Only Memory) type **56**,
- **[0072]** a Random Access Memory or RAM **57**,
- **[0073]** one or several I/O (Input/Output) devices **54** such as for example a keyboard, a mouse, a webcam, and
- **[0074]** a power source **58**.

[0075] The device **5** also comprises a display device **53** of display screen type directly connected to the graphics card **52** to display notably the displaying of synthesized images calculated and composed in the graphics card, for example live. The use of a dedicated bus to connect the display device **53** to the graphics card **52** offers the advantage of having much greater data transmission bitrates and thus reducing the latency time for the displaying of images composed by the

graphics card. According to a variant, a display device is external to the device **5** and is connected to the device **5** by a cable transmitting the display signals. The device **5**, for example the graphics card **52**, comprises a means for transmission or connection (not shown in FIG. **5**) adapted to transmit a display signal to an external display means such as for example an LCD or plasma screen or a video-projector.

[0076] It is noted that the word "register" used in the description of memories **52**, **56** and **57** designates in each of the memories mentioned, both a memory zone of low capacity (some binary data) as well as a memory zone of large capacity (enabling a whole program to be stored or all or part of the data representative of data calculated or to be displayed).

[0077] When switched-on, the microprocessor **51** loads and executes the instructions of the program contained in the RAM **57**.

[0078] The random access memory **57** notably comprises:

[0079] in a register **530**, the operating program of the microprocessor **51** responsible for switching on the device **5**,

[0080] calibration parameters **571** representative of intrinsic and extrinsic parameters of the acquisition devices used for acquiring the images **110** to **150** representative of the scene to be modelled;

[0081] data **572** representative of the texture associated with the images **110** to **150**, for example RGB information associated with the pixels of the images **110** to **150**;

[0082] data **573** representative of the mesh (for example the coordinates of the vertex of the mesh elements of the mesh) of a surface representative of the scene represented in the images **110** to **150**;

[0083] data **574** representative of the visibility of the mesh elements in the image **110** to **150** (for example a label visible and a label non visible associated with the pixels of the image and with the mesh elements of the mesh) when these data are available.

[0084] The algorithms implementing the steps of the method specific to the invention and described hereafter are stored in the memory GRAM **57** of the graphics card **52** associated with the device **5** implementing these steps. When switched on and once the parameters **570** representative of the environment are loaded into the RAM **57**, the graphic processors **520** of the graphics card **52** load these parameters into the GRAM **521** and execute the instructions of these algorithms in the form of microprograms of "shader" type using HLSL (High Level Shader Language) language or GLSL (OpenGL Shading Language) for example.

[0085] The random access memory GRAM **521** notably comprises:

[0086] in a register **5210**, the data representative of the texture associated with the images **110** to **150**,

[0087] data **5211** representative of a depth information (which is estimated or received with the texture information) associated with the pixels of the images **110** to **150**,

[0088] data **5212** representative of the mesh elements (for example coordinates of the vexels),

[0089] parameters **5213** representative of the visibility information associated with the mesh elements of the mesh, and

[0090] data **5214** representative of an error information associated with the candidate first texture information to be assigned to a given mesh element,

[0091] parameter **5215** representative of the selected first texture information among the plurality of candidate first texture information.

[0092] According to a variant, a part of the RAM **57** is assigned by the CPU **51** for storage of the values **5211** to **5214** and the parameters **5215** if the memory storage space available in GRAM **521** is insufficient. This variant however causes greater latency time in the composition of an image comprising a representation of the environment **1** composed from microprograms contained in the GPUs as the data must be transmitted from the graphics card to the random access memory **57** passing by the bus **55** for which the transmission capacities are generally inferior to those available in the graphics card for transmission of data from the GPUs to the GRAM and vice-versa.

[0093] According to another variant, the power supply **58** is external to the device **5**.

[0094] FIG. **6** illustrates a method for texturing a mesh of a surface **1** implemented for example in a computation unit illustrated on FIG. **5**, according to a particular and non limitative embodiment of the invention.

[0095] During an initialisation step **60**, the different parameters of the device **5** are updated. In particular, the parameters representative of the images and associated depth maps are initialised in any way.

[0096] Then, during a step **61**, for each first image of a set of first images, a first texture information is associated with the mesh element. A first image corresponds to an image of a plurality of images representing a scene according to several points of view, in which image a given mesh element is visible or at least partially visible. The visibility of the mesh element is determined by projecting it in the first images and comparing the depth of the mesh element (determined by converting the coordinates of the mesh element from the world space to the image space) with the depth information associated with the pixels of the images onto which the mesh element is projected. According to a variant, the visibility information associated with the mesh element and each image is received in addition to the data representative of the images (texture information, for example RGB information associated with the pixels of the images). As the mesh element is visible in at least two first images, there is several first texture information (called candidate first texture information) which is associated with the mesh element.

[0097] Then, during a step **62**, the candidate first texture information is each projected onto the first images. According to a variant, the candidate first texture information are each projected onto only a part of the first images, for example on all first images excepted the first image providing the candidate first texture information which is projected on the first images. A projection of a mesh element onto a first image corresponds to a conversion of the coordinates of a mesh element from the world space to the image space of the first image.

[0098] Then, during a step **63**, en error information is estimated for each candidate first texture information. The error information is computed according to the results of a comparison between the projected first texture information and the texture information associated with the first image on which is projected the candidate first texture information. The comparison is performed for each first image onto which the candidate first texture information is projected. According to a non limitative example, the error information associated with a candidate first texture information corresponds to the

sum of all comparison results between the candidate first texture information and the texture information associated with the first images onto which the candidate first texture information is projected. According to a variant, the error information associated with a candidate first texture information corresponds to the average of all comparison results between the candidate first texture information and the texture information associated with the first images onto which the candidate first texture information is projected

[0099] Lastly, during a step **64**, one of the candidate first texture information is selected among the plurality of candidate first texture information according to the error information estimated for and associated with each candidate first texture information. Advantageously, the selected first texture information corresponds to the candidate first texture information having the least value of error information.

[0100] Naturally, the invention is not limited to the embodiments previously described.

[0101] In particular, the invention is not restricted to a method for texturing a mesh but extends to the computation unit implementing such a method and to the display device or mobile device comprising a computation unit implementing the texturing of a mesh or the display of the images resulting from the texturing process. The invention also concerns a method for selecting a texture information to be assigned to a mesh element among a plurality of candidate texture information and also to a method for coding and transmitting the selected texture information associated with a mesh of a 2D or 3D representation of a scene in a multi-view system.

[0102] The implementation of calculations necessary to the texturing of the mesh and to the selection of a texture information to be assigned to a mesh element is not limited either to an implementation in shader type microprograms but also extends to an implementation in any program type, for example programs that can be executed by a CPU type microprocessor.

[0103] The use of the invention is not limited to a live utilisation but also extends to any other utilisation, for example for processing known as postproduction processing in a recording studio for the display of synthesis images for example. The implementation of the invention in postproduction offers the advantage of providing an excellent visual display in terms of realism notably while reducing the required calculation time.

[0104] The implementations described herein may be implemented in, for example, a method or a process, an apparatus, a software program, a data stream, or a signal. Even if only discussed in the context of a single form of implementation (for example, discussed only as a method), the implementation of features discussed may also be implemented in other forms (for example, an apparatus or program). An apparatus may be implemented in, for example, appropriate hardware, software, and firmware. The methods may be implemented in, for example, an apparatus such as, for example, a processor, which refers to processing devices in general, including, for example, a computer, a microprocessor, an integrated circuit, or a programmable logic device. Processors also include communication devices, such as, for example, computers, cell phones, portable/personal digital assistants ("PDAs"), and other devices that facilitate communication of information between end-users.

[0105] Implementations of the various processes and features described herein may be embodied in a variety of different equipment or applications, particularly, for example, equipment or applications associated with data encoding, data decoding, view generation, texture processing, and other processing of images and related texture information and/or depth information. Examples of such equipment include an encoder, a decoder, a post-processor processing output from a decoder, a pre-processor providing input to an encoder, a video coder, a video decoder, a video codec, a web server, a set-top box, a laptop, a personal computer, a cell phone, a PDA, and other communication devices. As should be clear, the equipment may be mobile and even installed in a mobile vehicle.

[0106] Additionally, the methods may be implemented by instructions being performed by a processor, and such instructions (and/or data values produced by an implementation) may be stored on a processor-readable medium such as, for example, an integrated circuit, a software carrier or other storage device such as, for example, a hard disk, a compact diskette ("CD"), an optical disc (such as, for example, a DVD, often referred to as a digital versatile disc or a digital video disc), a random access memory ("RAM"), or a read-only memory ("ROM"). The instructions may form an application program tangibly embodied on a processor-readable medium. Instructions may be, for example, in hardware, firmware, software, or a combination. Instructions may be found in, for example, an operating system, a separate application, or a combination of the two. A processor may be characterized, therefore, as, for example, both a device configured to carry out a process and a device that includes a processor-readable medium (such as a storage device) having instructions for carrying out a process. Further, a processor-readable medium may store, in addition to or in lieu of instructions, data values produced by an implementation.

[0107] As will be evident to one of skill in the art, implementations may produce a variety of signals formatted to carry information that may be, for example, stored or transmitted. The information may include, for example, instructions for performing a method, or data produced by one of the described implementations. For example, a signal may be formatted to carry as data the rules for writing or reading the syntax of a described embodiment, or to carry as data the actual syntax-values written by a described embodiment. Such a signal may be formatted, for example, as an electromagnetic wave (for example, using a radio frequency portion of spectrum) or as a baseband signal. The formatting may include, for example, encoding a data stream and modulating a carrier with the encoded data stream. The information that the signal carries may be, for example, analog or digital information. The signal may be transmitted over a variety of different wired or wireless links, as is known. The signal may be stored on a processor-readable medium.

[0108] A number of implementations have been described. Nevertheless, it will be understood that various modifications may be made. For example, elements of different implementations may be combined, supplemented, modified, or removed to produce other implementations. Additionally, one of ordinary skill will understand that other structures and processes may be substituted for those disclosed and the resulting implementations will perform at least substantially the same function(s), in at least substantially the same way(s), to achieve at least substantially the same result(s) as the implementations disclosed. Accordingly, these and other implementations are contemplated by this application.

1. A method for texturing a mesh, the mesh being associated with a surface representative of a scene captured accord-

ing to a plurality of points of view, an image of the scene comprising a texture information being associated with each point of view, wherein at least a mesh element of said mesh is at least partially visible from at least two first images of the plurality of images, the method comprising:

for each first image, association of a first texture information with said mesh element,

projection of the first texture information in said at least two first images,

for each first texture information, estimation of an error information according to a comparison result between the projected first texture information and the texture information of said at least two first images, the error information being representative of a difference between the projected first texture information and the texture information of said at least two first images, and

selection of one of the at least two first texture information according to said error information.

2. The method according to claim **1**, wherein the error information associated with a first texture information corresponds to a sum of the comparison results between the projected first texture information and each said at least two first images.

3. The method according to claim **1**, wherein the selected first texture information corresponds to the first texture information having the least error information.

4. The method according to claim **1**, wherein the method further comprises:

projecting said mesh element in at least a part of said images of the scene,

for each image of the at least a part of images, comparing a depth information associated with the projected mesh element and a depth information associated with the image,

for each image of said at least a part of images, determining a visibility information associated with said mesh element according to the result of the comparison of previous step, a mesh element being labeled as at least partially visible or not visible in said image according to said visibility information.

5. The method according to claim **4**, wherein said projection of said mesh element corresponds to a conversion of the coordinates of the mesh element from world space into the image space of the image.

6. The method according to claim **1**, wherein each image comprises a plurality of pixels, video information being associated with the plurality of pixels.

7. The method according to claim **1**, wherein each image is represented by using Multi-view Video plus Depth (MVD) representation.

8. A computation unit configured to texture a mesh, the mesh being associated with a surface representative of a scene captured according to a plurality of images, each image comprising texture information, wherein at least a mesh element of said mesh is at least partially visible from at least two first images of the plurality of images, the computation unit comprising at least one processor configured for:

associating, for each first image, a first texture information with said mesh element,

projecting the first texture information in said at least two first images,

estimating, for each first texture information, an error information according to a comparison result between the projected first texture information and the texture information of said at least two first images, the error information being representative of a difference between the projected first texture information and the texture information of said at least two first images, and

selecting one of the at least two first texture information according to said error information.

9. The computation unit according to claim **8**, wherein the error information associated with a first texture information corresponds to a sum of the comparison results between the projected first texture information and each said at least two first images.

10. The computation unit according to claim **8**, wherein the selected first texture information corresponds to the first texture information having the least error information.

11. The computation unit according to claim **8**, wherein the at least one processor is further configured for:

projecting said mesh element in at least a part of said images of the scene,

comparing, for each image of the at least a part of images, between a depth information associated with the projected mesh element and a depth information associated with the image,

determining, for each image at said least a part of said images, a visibility information associated with said mesh element according to the result of the comparison, a mesh element being labeled as at least partially visible or not visible in said image according to said visibility information.

12. The computation unit according to claim **11**, wherein said means for projecting said mesh element comprise means for converting the coordinates of the mesh element from world space into the image space of the image.

* * * * *