Image-adapted voxelization in multicamera settings

Jordi Salvador, Josep R. Casas
Image Processing Group
Technical University of Catalonia
Barcelona
{aljsal,josep}@gps.tsc.upc.edu

***Abstract*—This paper presents a new procedure to explore 3D space for visual analysis in multicamera environments. We propose a 3D geometry to sample (*voxelize*) the space when checking the consistency of the analyzed features in the multiple views. Contrary to regular voxelization, the proposed geometry is irregular in 3D, but becomes regular once projected onto camera images. The aim is to better exploit the redundancy and the enriched information provided in multiview frameworks at the analysis stage. This is accomplished by balancing the usage of the available data (i.e. captured pixels) across the multiple cameras, instead of focusing in a regular sampling of the 3D space from which we do not have direct data. An efficient recursive algorithm that uses the proposed geometry is outlined, and the experimental results reflect higher accuracy than regular voxelization with equivalent restrictions for the chosen multiview analysis applications.**

***Keywords*—camera calibration; epipolar line; multicamera; multiview; non-regular voxelization; visual analysis**

***Topic area*—Multimedia Processing (visual, multiview, 3-D geometry)**

I. Introduction

Visual analysis in multicamera environments provides enhanced robustness by exploiting redundancy in visual information from the different images of the same scene. Multiview approaches for foreground detection, space carving, face/person detection and tracking, gesture analysis, etc. benefit from common information present in the multiple views by checking the consistency of the analyzed primitives (color, foreground, salient points…) from the (redundant) available projections of the actual 3D scene. In addition, the non-redundant information extracted from the multiple views, enriches the visual analysis process with 3D cues, which disambiguate occlusions and provide visual information from otherwise occluded parts of the scene.

Multiview redundancy enhances analysis robustness by checking consistency among images captured by multiple calibrated cameras. One often resorts to an ordered scanning of the 3D space (as in [1]). Spatial regions of the 3D space are sequentially analyzed from their projections in the multiple cameras. The usual approach is regular voxelization, where the working 3D space is sampled at regularly spaced intervals in its orthogonal axes, giving rise to elementary cubes (voxels). The problem arises from the fact that the sampling geometry generated by this particular scanning of the 3D space is distorted by the projection onto the cameras. Once projected onto the camera images, the sampling geometry becomes irregular, and the amount of available data (pixels) used to analyze each space region (voxel) depends on its distance to the camera and the intrinsic camera parameters. This can be shown in two different examples:

- When projecting two separated regular voxels (equally sized 3D volumes in space) on the same camera image, the amount of pixels used for visual analysis can be quite different when the voxels are located at different distances from the camera.
- When projecting the same regular voxel on two different cameras, the amount of pixels in the projections is also quite varying. An illustration of this case can be seen in Fig. 1.

We have evaluated this problem in a particular situation. A sampling geometry with cubic voxels of size 3 cm is defined in a Smart Room equipped with 5 cameras (four cameras placed on the corners and the fifth one ceiling mounted as a zenithal cam). We compute the statistics of the projection size (in pixels) of all the voxels in the 3D working space. The histogram of the voxel projection size is shown in Fig. 2 for one of the corner cameras. Table I outlines mean and dispersion values of the voxel projection size for all the cameras, and reveals that the dispersion of the projection sizes is really high (the same order as the mean value).

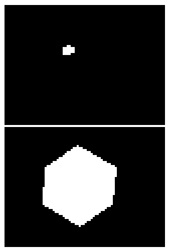

Figure 1. The same voxel in 3D space seen by a close camera and by a far one

The situation described is not favorable when checking the consistency of the analysis in the multiple projections of the actual 3D scene. We are using quite different amounts of data (the number of projected pixels for each voxel in each view) to analyze each individual regions in 3D space. We refer to this situation as "3D space voxelization **non-adapted** to the image data" in multiview analysis.

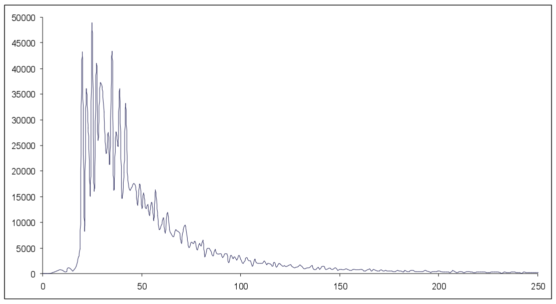

Figure 2. Histogram of the voxel projection size on a calibrated camera (in pixels) of a 3 cm side voxel in a 4x5x2 m^3 room

TABLE I. MEASUREMENTS OF VOXEL PROJECTION SIZES IN PIXELS FOR 3 CM SIDED VOXELS

	Minimum projection size	*Maximum projection size*	*Mean projection size*	*Dispersion projection size*
Camera 1	4	1797	60	67
Camera 2	4	1864	58	65
Camera 3	4	1650	59	66
Camera 4	4	2155	60	69
Camera 5	2	296	40	22

One way of taking into account such lack of adaptation to the image data, is to avoid giving the same importance to each view in the consistency check for every voxel. An alternative strategy is the one we present in this paper.

The aim of the present paper is to define a new approach to perform multiview analysis using a 3D space scanning procedure better **adapted** to the image data; that is, without such a dispersion in the amount of data used for the consistency check at each 3D space unit. The reasoning behind this approach is that the actual data we have available for analysis are the projected images, and it does not make much sense to scan the 3D space (from which we do not have direct measurements) with a regular geometry, while this results in a non-regular geometry when projected on the camera images. Please bear in mind that our actual visual measurements (pixel data) come from the projections in the camera images, as opposed to data coming from 3D measurements (as MRI, PET, etc), where the sensors scan directly the 3D space.

Therefore, we change the paradigm trying to adapt the visual analysis to the available data. Instead of scanning the 3D space with regular voxels, from which there is not any direct data, the camera images are divided in a regular way, and the 3D equivalents of such divisions are used as the scanning geometry. In this way, the defined geometry is adapted to images.

The following section in this paper explains in higher detail how the new scanning geometry is defined and how it is used to scan the 3D space in a recursive algorithm. The next section presents the results obtained for a selected analysis technique and compares them to an existent approach (regular voxelization). Finally, the main advantages and disadvantages of the new method are discussed in the last section.

II. PROPOSED METHOD

In this section we present the tools we use to obtain a geometry adapted to images for 3D space scanning. First, we define the *quadrant* as an image region. Then, the *cone* is obtained as the back-projection of the *quadrant*. The *cone* is the intermediate step to compute the final image-adapted geometry, which is based in what we call *conexels*. Finally, we outline a recursive algorithm for 3D space scanning in multicamera settings, which has proven useful for several analysis techniques.

A. *Image regions:* ***quadrants***

As stated above, one of the principal aims of this method is to avoid dispersion in the amount of data used in analysis. To achieve that, images are divided in quadrants for scanning. Quadrants are defined as square shaped, non-overlapping regions in the projected images. 3D space scanning will be defined based in the geometry generated by the quadrants, instead of using the voxel-based geometry. The expected behavior of such approach is that it will use the same amount of data in every image when scanning a 3D space region: their projections will always lie inside the selected quadrants.

B. *Back-projection of quadrants in 3D space:* ***cones***

To obtain the 3D back-projection of a quadrant in an image, the back-projected ray of every corner of the quadrant must be computed, as shown in [2]. Combining those 4 ray equations, 4 plane inequations can be defined so the pixels in the quadrant are the projection of the inner volume enclosed by the four planes. The *Center Of Projection* (COP) of the camera is the main vertex of the cone, which results in a pyramidal shape without a basis. An illustration of one of such cones can be seen in Fig. 3.

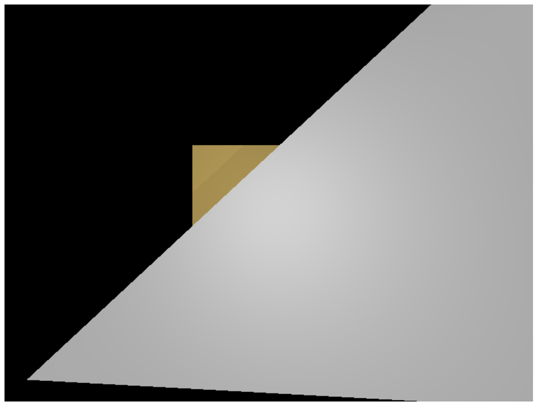

Figure 3. Back-projected cone from a camera positioned in the bottom left corner of the image as seen from a zenithal camera. The floor of the room (colored square in the background) is included as a reference

C. *Intersection of two or more cones:* ***conexels***

We define the elementary 3D space volume (*conexel*) of our scanning geometry adapted to images in the following way:

1) Choose a quadrant for every available image

2) Compute the back-projected cone for every quadrant

3) The conexel is then defined as the intersection of the back-projected cones of every quadrant

The name conexel comes from "cone element."[1] In Fig. 4, a 3D view of a conexel obtained as the intersection of 2 cones from quadrants in 2 camera images is shown.

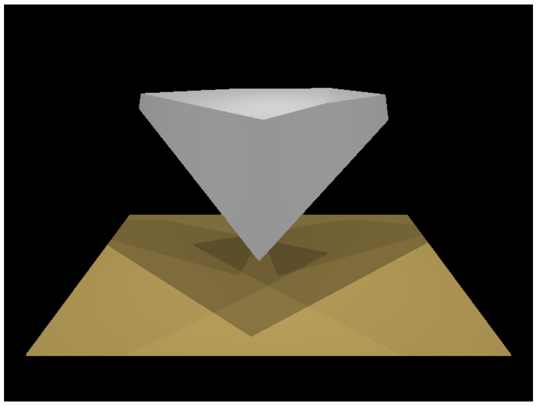

Figure 4. Conexel obtained as the intersection of the back-projected cones from 2 cameras in the room corners. A floor reference is again shown in the picture

The projection of a conexel in every camera can be computed in a fast way, without having to compute its real 3D geometry and project it on every camera image. This is accomplished by selecting the inner zone to all the epipolar lines (refer to [5]) of the corners of the quadrants on the other cams computed using *fundamental matrices,* which relate points in source camera images with rays in target cameras. Changing equations of the epipolar lines by inequations, we can define two image regions: pixels which lay inside the cone projection and pixels which lie outside it. Pixels inside the current quadrant for the working camera must be checked, and only those which also lay inside all cone projections are selected as part of the projection of the conexel on the camera image.

Please note that, when projecting again the conexel obtained by cone intersection onto the camera images, the projections do not completely cover the quadrants which generated the conexel. Therefore, there will still be some dispersion in the amount of pixel data used for the analysis of the conexel, which is now our elementary scanning volume in 3D space. Anyway, the dispersion is expected to be smaller than with the regular voxelization approach. In this sense, our proposal is better adapted to the image data. In fact, the number of pixels of the projection of the conexel in each camera view will range from 1 to the total number of pixels in the quadrant. This dispersion range is usually much smaller than the dispersion range with regular voxelization, as seen in the introduction.

D. *Recursive scanning algorithm*

The presented approach can be easily introduced in a recursive 3D space scanning algorithm, which performs as a quad-tree decomposition at the level of the projected images. We recursively divide each quadrant in four quadrants (depending on the consistency check) and we carry on dividing the resulting quadrant in four sub-quadrants. For each division, the result is a new set of conexels which always lie in the previous larger one. This result can be used to selectively enhance the resolution of 3D analysis in places where it is needed, such as objects contours, without using the highest resolution in places where it is not necessary. Thus, a space scanning algorithm based on the proposed procedure may start with color consistency checks at rough resolution (large conexels) and progressively refine it by recursive subdivision of only those conexels where needed. An *m-tree* ([6]) is used to store the results. Its implementation includes a set of functions which allow moving up, down and to the sides in its structure. As the number of cameras may vary, the algorithm performs loops without hard-coding them on a fixed number of cameras, defining a vector to store the current chosen quadrant for every camera and a variable to store the current camera in use.

To sum up, the main recursive algorithm to scan 3D space based on the geometry defined by the conexels works as follows:

1. Set the current camera to zero (first camera) and the chosen quadrant vector to all zeros.
2. If the currently selected camera index is larger or equal to zero go to the next step, if not finish.
3. For all cameras with index smaller than the currently selected camera, obtain cone projections for selected quadrants on other cams onto the currently selected camera image, and cone projections for the current quadrant in the currently selected camera onto the previous camera images. If any of the cameras does not have any pixel belonging to the conexel projection, it means that no conexel exists for the current set of quadrants. In that case jump to step 6, if not go to the next step.
4. If the currently selected camera is the last available one (meaning that a conexel exists for the current set of quadrants), count the number of pixels of the conexel projection on every camera and, if it is different from zero, go to the next step. In any other case select next camera and jump to step 2.

[1] In analogy with pixel from "picture element" and "voxel" from "volume element"

[3] The available quadrants in every camera are stored in the *m-tree*

5. At this step **any analysis function** can be implemented for a multiview consistency check on the projected pixels corresponding to the obtained 3D region. In case that the consistency check needs a higher resolution jump to step 7, if not go to the next step after storing the results in the *m-tree*.
6. If the current quadrant in the current camera is not the last, increment it. In other case, set it to 0, decrement the currently selected camera index and repeat this step if the current camera is bigger or equal than zero. Finally jump to step 2.
7. Subdivide each quadrant in new smaller quadrants in every camera, go down in the *m-tree*, call recursively this procedure and go up in the m-tree again[3]. Then jump to step 6.

III. Experimental Results and Discussion

As stated in the introduction, an improvement of analysis results is expected due to the fact that the presented scanning approach minimizes the dispersion in the number of pixels used for each consistency check of an elementary volume projected onto the different camera images. This section provides a proof of concept aiming to validate to which extent the proposed geometry improves analysis applications in multicamera settings.

The application we have chosen as a proof of concept is 3D foreground segmentation (shape-from-silhouette in [3]). In its real usage the input data are black and white foreground segmentation masks obtained by 2D foreground segmentation algorithms (like [4]) from the camera projections. This presents some problems like misses or false detections and the presence of noise in form of isolated or shortly grouped foreground pixels. For the proof of concept, we have generated 5 simple synthetic scenes. A sphere with a diameter of 1 meter generated with 100 vertices is placed in 5 different positions inside the working space in the room. This foreground constitutes the ground-truth, because the projections are generated taking into account the intrinsic and extrinsic parameters of every camera. An example of one of these input projections can be seen in Fig. 5.

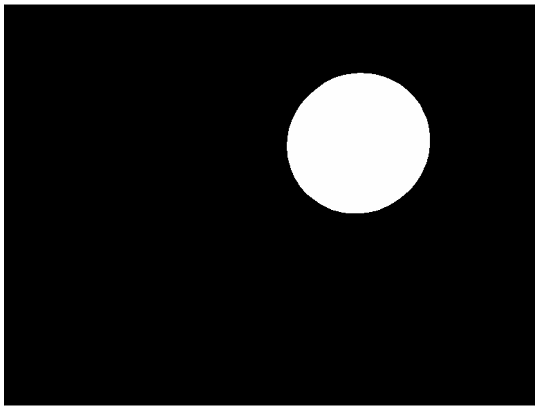

Figure 5. Input projection for camera 2 for the first of the 5 generated scenes. It doesn't contain noise effects, nor misses or false detections

We compare our scanning approach for the 3D foreground segmentation algorithm with regular voxelization as the competing method. The criteria to compare results is using a voxel side that, in average, has a projection size in pixels on camera images equal to the number of pixels of the smallest quadrant used in the reconstruction. The formula

$$nPix \approx \pi r^2 \approx \pi\left(\sqrt{3}/2 \cdot voxSide \cdot K_0 / dist\left(voxCenter, COP\right)\right)^2 = \alpha \cdot voxSide^2$$

allows computing an equivalent voxel side from the number of pixels we want the voxel to be projected to. We approximate the average voxel projection size in pixels by the projection size of the central voxel in the sphere, and compute the α in the formula for every scene and for every camera. Then, an average of α is computed for every scene along all the available cameras and, when setting the number of pixels to 9, the equivalent voxel side in all scenes is around 1.1 cm so we take 1 cm as the equivalent voxel side.

After performing the analysis with both methods, the 3D foreground volume information obtained is projected on for all the cameras and a distance function with regards to the original ground truth is computed for every available image. The distance function used is

$$\mathrm{dist}(rec, gt) = (\mathrm{area}(rec \cup gt) - \mathrm{area}(rec \cap gt)) / \mathrm{area}(gt),$$

that is, the number of different pixels among reconstructed masks and ground-truth ones divided by the number of pixels of the ground-truth. The results for the 5 scenes and the 5 cameras are listed in Table II for regular voxelization and in Table III for image-adapted voxelization. A factor of 100 has been applied to make results more easily readable.

TABLE II. Distance to Ground-Truth (in %) for Regular Voxelization

	Scene 1	*Scene 2*	*Scene 3*	*Scene 4*	*Scene 5*
Camera 1	7.40	7.28	7.65	7.57	7.46
Camera 2	7.25	7.46	7.62	7.74	7.34
Camera 3	7.68	7.63	7.46	7.24	7.52
Camera 4	7.62	7.65	7.39	7.49	7.35
Camera 5	7.31	7.46	7.35	7.38	7.47

TABLE III. Distance to Ground-truth (in %) for Image-Adapted Voxelization

	Scene 1	*Scene 2*	*Scene 3*	*Scene 4*	*Scene 5*
Camera 1	2.03	2.55	3.12	2.91	2.56
Camera 2	2.51	1.95	3.13	2.98	2.83
Camera 3	3.29	2.97	1.89	2.37	2.76
Camera 4	2.74	3.19	2.50	2.01	2.81
Camera 5	2.36	2.37	2.66	2.74	2.52

To depict those results, Fig. 6 shows in white pixel differences between the ground-truth image and the projection from the reconstructed 3D object for the first scene projected on camera 2 for regular voxelization. Fig. 7 is the equivalent for image-adapted voxelization. Please note how the dispersion in the amount of data used for analysis in regular voxelization causes larger false volumes to appear.

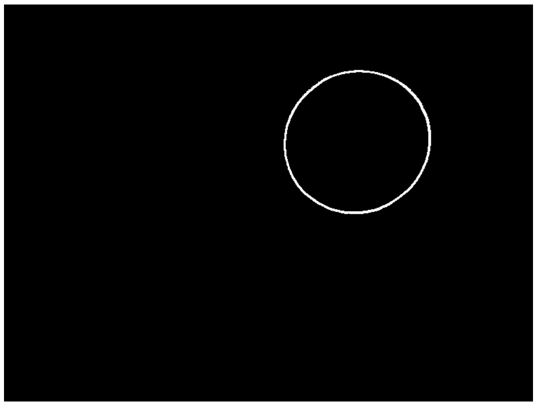

Figure 6. Pixel differences between reconstruction with regular voxelization and ground-truth for the first scene on camera 2

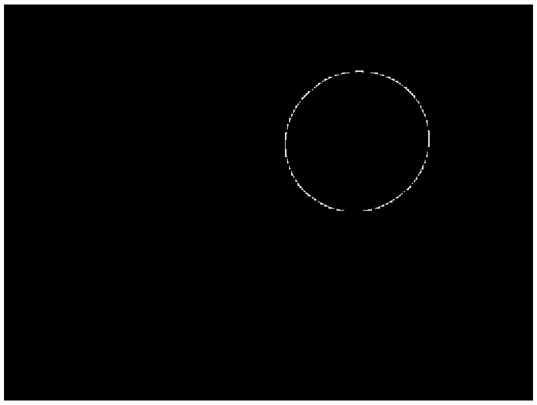

Figure 7. Pixel differences between reconstruction with image-adapted voxelization and ground-truth for the first scene on camera 2

IV. Conclusions and Future Work

We have presented a multiview analysis approach using a 3D space scanning procedure which is adapted to the images. Instead of exploring 3D space (from which we do not have direct measurements, but only projections) with regular geometry, the proposed procedure defines a geometry based on image quadrants. This avoids dispersion in the amount of data used for the consistency check, which is a fundamental step to exploit redundancy in visual analysis from the multiple views.

Furthermore, a recursive algorithm based on such method has been implemented and proven to work. The results obtained show less dispersion and increased spatial precision when compared with regular voxelization, as expected because of the balanced usage of the directly measured data.

As a problem, we must remark that dispersion in the amount of data used in analysis has not been completely cancelled, but it is more controlled than with regular voxelization techniques.

The main directions for future improvements must focus in the study of conexels' connectivity and how to use it to remove inner conexels from analysis results in case of need. Another field of study is the set of situations in which conexels are defined from a smaller number of cameras to deal with cases where a conexel is only visible in a subset of all the available cameras.

References

[1] K. M. Cheung, T. Kanade, J.-Y. Bouguet, and M. Holler, "A real time system for robust 3d voxel reconstruction of human motions," in *Proceedings of the 2000 IEEE Conference on Computer Vision and Pattern Recognition* (CVPR '00), volume 2, pages 714 – 720, June 2000. 4, 10

[2] O. Garcia and J.R. Casas, "Functionalities for mapping 2D images and 3D world objects in a multicamera environment," *6th International Workshop on Image Analysis for Multimedia Interactive Services* (WIAMIS), Montreux, Switzerland, April 13-15, 2005

[3] J.L. Landabaso and M. Pardas, "Foreground regions extraction and characterization towards real-time object tracking," in *Proceedings of Joint Workshop on Multimodal Interaction and Related Machine Learning Algorithms* (MLMI '05), 2005

[4] C. Stauffer, W.E.L. Grimson, "Learning patterns of activity using real-time tracking," *IEEE Trans. on Pattern Anal. and Machine Intel.*, 22(8):747–757, 2000.

[5] Y. Ma, S. Soatto, J. Kosecká and S. S. Sastry, *An Invitation to 3-D Vision*. New York: Springer-Verlag, 2004

[6] Lu, T, *The Enumeration of Trees with and without Given Limbs.* Disc. Math. 154, 153-165, 1996.