GEOCAST: UNIFYING DEPTH VIDEO WITH CAMERA META-DATA

Gernot Ziegler, Lukas Heidenreich, Marcus Magnor and Hans-Peter Seidel

MPI Informatik, Saarbrücken, Germany

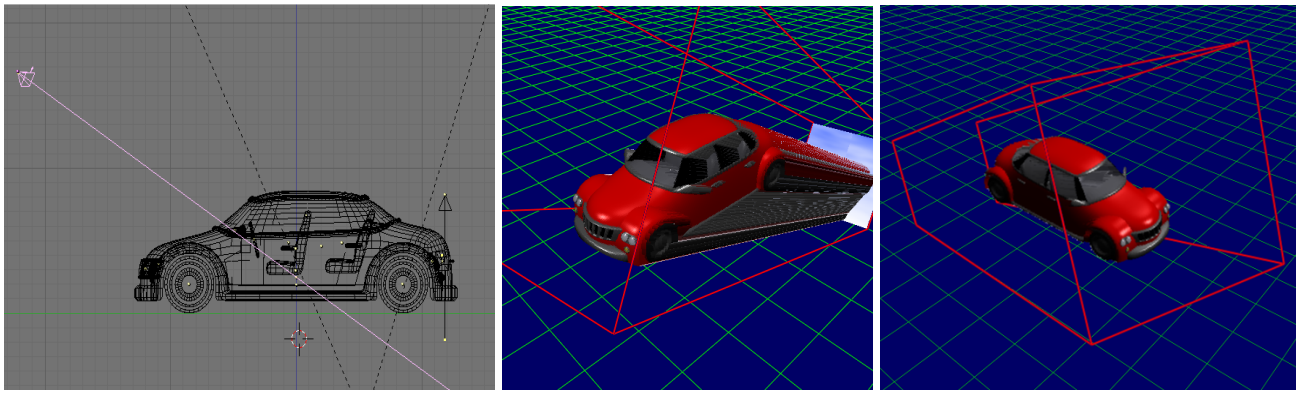


Fig. 1. Visualization of a GeoCast stream, which holds a dynamic camera view of a 3D model.
From left to right: 3D model and camera. A view without background clipping.. A view using Z-based clipping.

ABSTRACT

We present a storage format for placing multiple, dynamic 2.5D point sample streams (e.g. RGBZ video) or color video projections into the context of a common world space. The approach utilizes the concept of projective geometry to let virtual projectors "cast" the data from where it was originally recorded by the involved cameras. We exemplify the data format's versatility by demonstrating how several moving cameras reproduce 3D geometry exported from modeling software, and outline the extension to real-world acquisition. We also explain how this format can act as common interchange format for the camera parameters of lightfield/multi-view video footage.

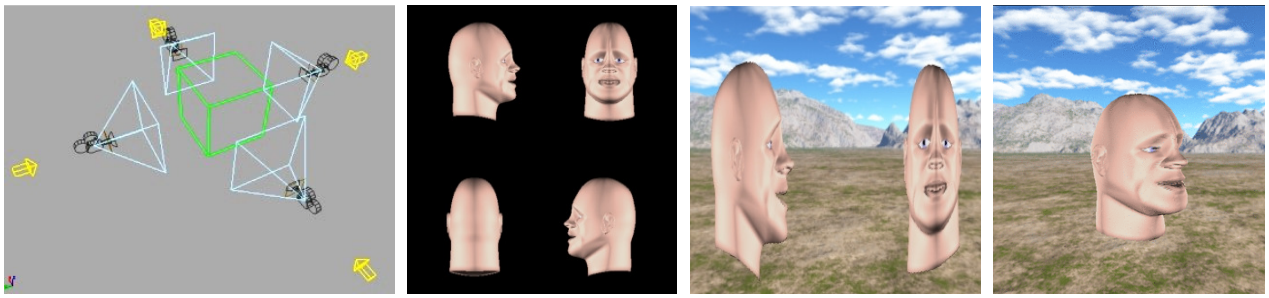
1. INTRODUCTION

With dropping prices for imaging sensors, it has become more and more interesting and viable to create multi-camera acquisition setups. Lightfield arrays and multi-view video setups all aim at recording a common scene from several simultaneous views. Presently, first prototypes of depth recording cameras are available ([1], [2]), and they, too, will soon be put into multi-view configurations.

As soon as multi-view data is available, it is important to know how the recorded camera views correlate to each other. Computer vision defines all the mathematical tools necessary to describe the mapping between image pixels and the recorded light rays that entered the camera [3]. Likewise, the description of this light ray mapping can be used to determine how an image with depth information (RGBZ image) maps into world space.

However, there is currently no *standardized* way to describe such multi-view camera setups, especially if dynamic camera movement is involved. Most often, visualization and post-processing developers need to communicate intensely with the footage producers to determine how the video footage can be projected back into world space to re-generate the original ray field, see [7] (or, in case of RGBZ video, to reconstruct the 2.5D geometry patches seen by the depth cameras).

We present a data representation that enfolds the video footage and a set of intrinsic and extrinsic camera parameters, and lets us re-create the world space scene as it was seen by the cameras. We exemplify how multi-view video footage, lightfield footage and multi-view RGBZ video can be converted into this representation and thus be exchanged with others without possibly ambiguous textual descriptions of the camera setup. We also demonstrate screen shots of a prototype rendering system that visualizes GeoCast based RGBZ video in its original world space setting.



(a) (b) (c) (d)

Fig. 2. Real-time fusion of GeoCast based RGBZ videos.
(a) Camera setup, stored in metadata files. (b) The four camera views (depth data not shown).
(c) Arranged images according to camera setup, background removed. (d) Final, merged view.

2. OVERVIEW

The explanation starts with virtual and real multi-view camera setups and the involved imaging parameters. Then, we lay out how multi-view video is structured inside GeoCast, and how different projection types are stored in the meta-data. RGBZ video receives special attention as its depth channel requires additional formatting.
Following that, we outline the major application areas: *Data exchange* between acquisition and image processing, *decoupled visualization* of image processing output and *direct visualization* from RGBZ multi-view video footage.
Finally, we address unresolved issues, the format's extension to unrectified footage and sketch how data compression fits together with GeoCast.

3. MULTI-VIEW CAMERA REPRESENTATION

As soon as a viewer of video footage is enabled to diverge from the original viewpoint of a camera, it is necessary to know how the image pixels map to the original light rays from the outside world, or, in case of a depth camera, how the depth samples map to the original geometry surface in world space. To this purpose, the image data needs to be bundled with camera metadata. In case of dynamic camera movement or intrinsic camera changes (e.g. zooming), this camera metadata actually is frame-specific, and thus needs to follow along with each video frame.

The most general way for representing this mapping is to use a textual description as exemplified in [3], which binds intrinsic and extrinsic parameters into one ray mapping procedure. However, most of the time, footage rectification (which removes the camera lenses' intrinsic distortions) happens before the footage is handed on to image matching or lightfield visualization software. Therefore, we omit intrinsic information (but present an outlook in the final section on how it can be included if needed) and assume that the image or surface projection into worldspace can be described by two 4x4 matrices, the *projection* matrix and the *modelview* matrix.

Traditionally, the coordinate system for the image to world space mapping is agreed upon between the footage's creators and its consumers, caused by the involved intrinsic parameters and image resolution.

Since we handle intrinsic parameters *before* the projection and GeoCast should encompass all possible image resolutions, we decided to apply the widely used and resolution-independent OpenGL coordinate system in our data representation and rely on a fast coordinate mapping from OpenGL's (x,y) floating-point pixel addressing domain of ([-1, 1][-1,1]) instead of the image's integer addressing ([0, xres], [0, yres]). See also [4] for more details.

The depth samples are represented in the same range as the OpenGL normalized device coordinates, where -1 refers to the near clipping plane, and +1 to the far clipping plane. . The 4x4 matrix transforms the depth values as well during the projection, such that the RGBZ values can generate the original surface in world space.

The advantages are: easy generation of synthetic footage, specified behaviour for depth samples, and simplified communication with visualization software developers. As shown later, we also lend other data format concepts from OpenGL to avoid lengthy and proprietary specifications.

Therefore, the main components that remain to be specified are the camera's imaging model and the actual camera positioning.

Pinhole camera: In real-world footage, this is the only possible imaging model. Among the available parameterizations, we chose to represent such cameras with their field of view, the image aspect ratio and their near and far clipping plane. The parameters map directly to OpenGL's `gluPerspective()` call.

Orthogonal camera: Space carving methods and other voxelization methods usually divide world space into uniform 3D grids. Image slices from such data volumes and synthetic RGBZ video from 3D geometry are candidates for this simple image-to-world mapping.

The parameters specify a horizontal and vertical window size and a near and far clipping which is handled by the OpenGL `glOrtho()` call.

General 4x4 projection matrix:

Sheared frusti, like the ones used in lightfields to resample onto a common plane, cannot be represented in the models above. However, they can still be written as a 4x4 projection matrix. Therefore, GeoCast allows the direct specification of an OpenGL-like `GL_PROJECTION` matrix.

ModelviewMatrix (Camera placement):

Image reconstruction from multi-view video requires knowledge on the original camera arrangement. Therefore, inter-camera calibration algorithms like [5] provide a correlation matrix between the different camera views. GeoCast expects the camera's positions and rotations in a 4x4 world to camera matrix transform. This way, we bypass the problems that a succession of translations and rotations produce (the need to define the order of operations, and gimble lock). Furthermore, we apply OpenGL definitions of a „camera" coordinate system even here: The negative Z axis points to where the image will be projected, and positive Y is the direction of the up vector. Any wide baseline or lightfield camera arrangement imaginable should fit in this general matrix.

4. STORAGE STRUCTURE

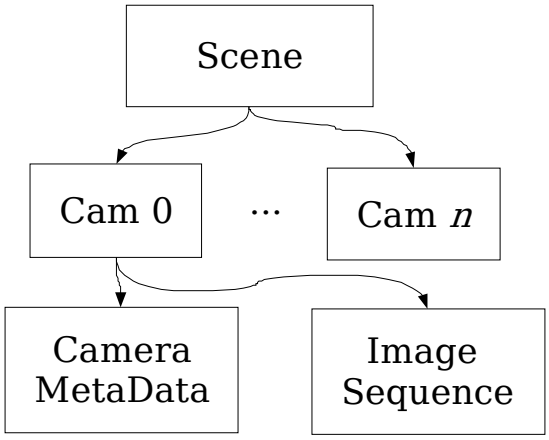


Fig. 3. General storage structure.

GeoCast is currently not intended as a compression format. Its main purpose is to simplify data exchange. Therefore, we aimed at easy read/write access and ease of implementation in its design to allow for fast adoption in research software. See Fig. 3 for a graphical overview.

The Scene file binds together all involved GeoCast streams. It also dictates the number of frames in this multi-view video sequence. Since the camera positions are stored in the GeoCast streams, they don't need to be specified in the scene file.

Each GeoCast stream consists of one or several MetaData files, which define the extrinsic camera parameters, and the video sequence itself.

The Camera MetaData contains the camera setup parameters as described in the previous chapter.

The video sequences of all cameras must have the same length. Presently, the video is stored as a sequence of image files for easy access.

The image format in itself is not specified, and the choice depends on the contributing application's I/O capabilities. Our own experimental software uses OpenEXR, a multi-layered floating point format introduced by Industrial Light & Magic [6], to store float-valued RGB image data together with a depth channel Z. OpenEXR features decent intra-frame data compression, and its multi-layered nature allows us to add additional, experimental data channels without affecting the basic RGBZ data storage.

5. APPLICATION AREAS

GeoCast's foremost purpose is to unify data export from diverse data sources, such as lightfield arrays, depth cameras and multi-view video recording setups.

Some of the possible application areas are:

Data exchange: Footage from the mentioned data sources usually needs conversion to fit the consuming image processing and computer vision software I/O. We provide GeoCast Camera MetaData for the KungFu girl data set and the Stanford CD data set to exemplify the new, unified camera setup description for multi-view and lightfield footage (available on request). We further provide a Blender software plugin prototype to export synthetic GeoCast streams from 3D models to assist multi-view video research (see Fig. 1, available on request).

Decoupled visualization: Stereo vision software usually needs to implement its own 3D renderer to visualize the reconstructed point cloud in world space. Our internal software prototypes export GeoCast files with RGBZ data instead, to decouple point cloud visualization from the vision tasks (see Fig. 4).

Direct visualization: Depth cameras hold the promise of providing real-time free viewpoint video at the consumer side without the need for intense post-processing. While we cannot yet provide real-world footage to demonstrate this, we can already now visualize multi-view RGBZ videos from cameras sweeping over synthetic 3D models to demonstrate the data format's working principle (see Fig. 1 and 2).

6. CONCLUSIONS AND FUTURE RESEARCH

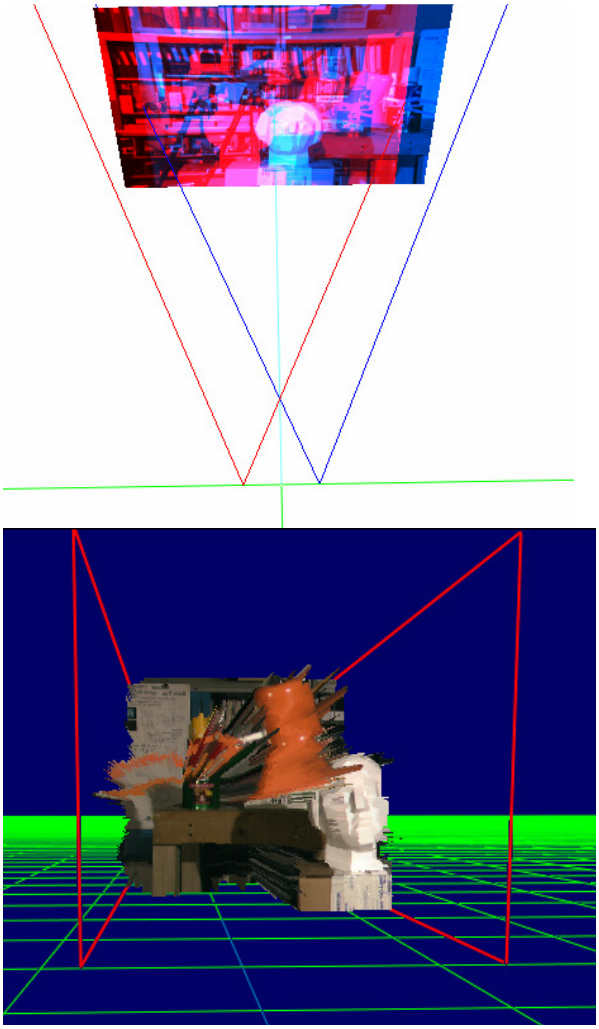


Fig. 3. Usage of GeoCast in stereo vision.
Top: GeoCast MetaData defines camera setups for reprojection of stereo images (red image: left eye, blue image: right eye), whereafter image processing commences.
Bottom: Visualizing a part of the reconstructed disparity map as RGBZ point cloud in world space.

Aside from its initial applications in RGBZ object reconstruction for free viewpoint video research, we now see potential use of GeoCast for multi-view video research in general, and hope to have proven this by describing some of the application areas.

In the future, several extensions to the format can be thought of:

Unrectified footage could be stored together if the camera's intrinsic lens parameters are present. This would allow image processing software to rectify the footage on its own. A future extension to the metadata can accommodate this supplemental information.

Data sources might also want to specify the *visibility of camera images* depending on the user viewpoint in the scene. We have e.g. investigated the specification of a viewing cone for RGBZ video to allow early culling in the rendering process. The same could apply to reprojected RGB video.

Segmentation hints could be useful for image processing, and be stored in additional OpenEXR layers.

Data compression is mostly a neglected topic in GeoCast, as the format mainly is targeted at temporary data exchange in research. Instead, we intend to cooperate with major standardization groups (e.g. MPEG 3DAV inside the MPEG consortium [8]) and other research groups (e.g. [9]) to embed the format's camera concepts into forthcoming multi-view video formats.

7. ACKNOWLEDGEMENTS

We would like to thank the European Commission for supporting our research in the context of grant 511568, FP6, under the acronym "3DTV".

8. REFERENCES

[1] 3DVSystems, http://www.3dvsystems.com , 2003.

[2] 3D-IP, http://www.3d-ip.com , 2004.

[3] Heikkilä & Silvén, "A Four-step Camera Calibration Procedure with Implicit Image Correction". IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'97), San Juan, Puerto Rico, p. 1106-1112.

[4] OpenGL 2.0 specification, http://www.opengl.org/documentation/spec.html , 2004.

[5] Ihrke, Ahrenberg and Magnor, "External camera calibration for synchronized multi-video systems", Proceedings of WSCG 2004.

[6] Industrial Light & Magic, http://www.openexr.com, 2003.

[7] MPEG-3DAV, "Validation Data for Camera Parameters of MVC-CfP", https://www.3dtv-research.org/3dav_Validation_Data

[8] MPEG-3DAV, http://www.chiariglione.org, 2005.

[9] Kauff, Smolic, Eisert, Fehn, Müller and Schäfer. „Data Format and Coding for Free Viewpoint Video". Proc. of IBC, Amsterdam, The Netherlands, Sep. 2005.