A Generalized Light-Field API and Management System.

Joan Blasco, Miguel Escrivà, Paco Abad, Ricardo Quirós,

Emilio Camahort, and Roberto Vivó

Computer Science Department.

Polytechnic University of Valencia.

Contents

- Introduction.
- Previous Work.
- System.
- API Design.
- Implementation.
- Results.
- Conclusions and Future Work.

Introduction

- Image-Based Rendering
  - Simple acquisition
  - Realistic representation
  - Rendering complexity depends on output image complexity.
- A light field represents the radiance flowing through all the points in a scene in all possible directions

Introduction

- We present a light-field modeling system
  - Efficiently build, store, combine, resample, retrieve and render LFs.
- Our Goals:
  - Compare representations, combine, and display on autostereoscopic displays.
  - Achieve interactive framerates.
  - Intuitive API.

Previous Work

- Light Field.
  - Plenoptic function [Adelson and Bergen91]
- Different parameterizations.
  - Planar anisotropic [Levoy96][Gorter96]
  - Spherical quasi-isotropic [Camahort98]
  - Unstructured [Buhler01]

Previous Work

- Levoy and Hanrahan and Gortler et al.
  - Planar Anisotropic Light Fields
  - Based on the two-plane parameterization.
  - Discretize the light-field by imposing rectilinear grids on both planes.
- Camahort et al.
  - Spherical quasi-isotropic.
  - Based on a spherical parametrization.

System

- Support for light-field modeling and rendering.
  - Different parameterizations.
  - Multiple input camera arrangements.
  - Different resolutions.
  - Different capture robot configurations.
  - Several storage strategies.
  - Multiple rendering algorithms.
  - Multiple display devices.

System

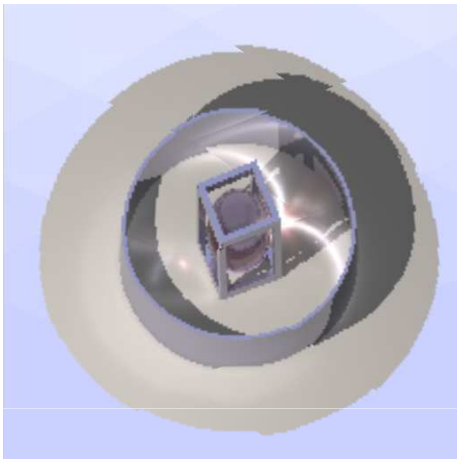
- Light-Field Representations.
  - Support drawing, generating,capturing and storing different light-field representations.
- Our software
  - Easily configurable.
  - Extended using specific lightfield plugins.
    - Spherical LF plugin
    - Planar LF plugin

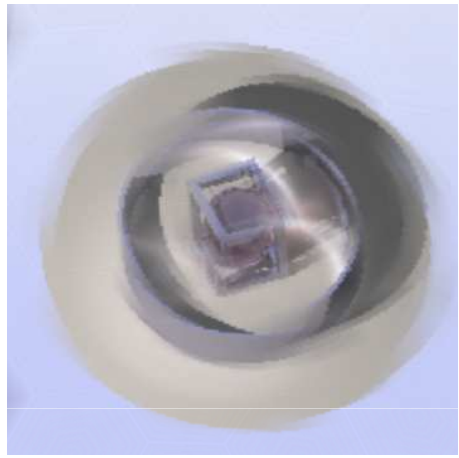
System

- Generation and Capture
  - Specific to each type of representation.
  - Camera iterators.
- Synthetic models
  - OpenGL, POV, Blender
- Real-world objects.
  - Camera mounted on a robotic arm

System

- Light-field's drawbacks
  - Huge storage req's
  - Artifacts due to discretization errors
    - Seams and ghosting
- Solution
  - Use of depth information.



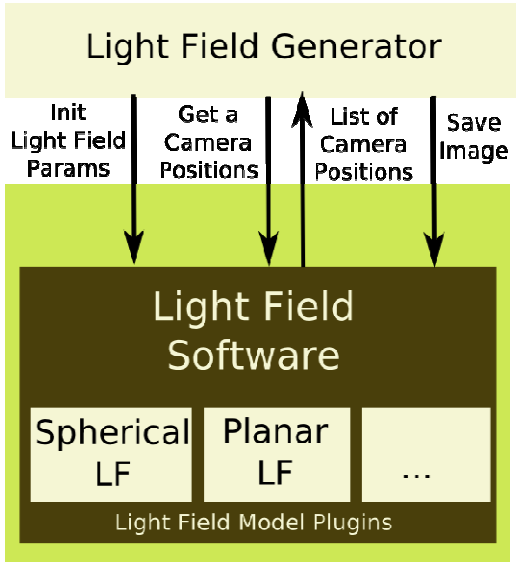


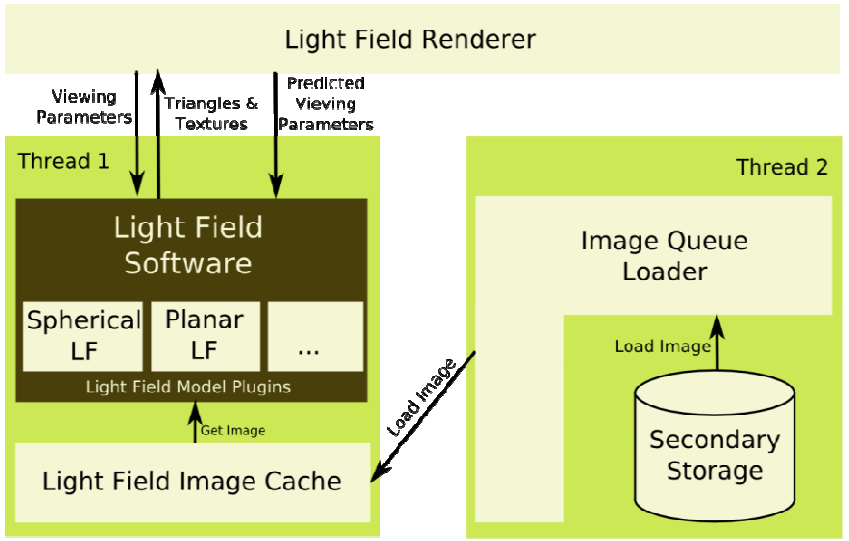
System

- Light-field composition supports
  - Managing different light-field models.
  - Integration with geometric information.
    - Labels, tags, regular objects,…
- Multiple Light Fields
  - Storing multiple images per directional sample.
- Correct renderings
  - Draw in proper back-to-front order.
  - Use depth information.

API Design

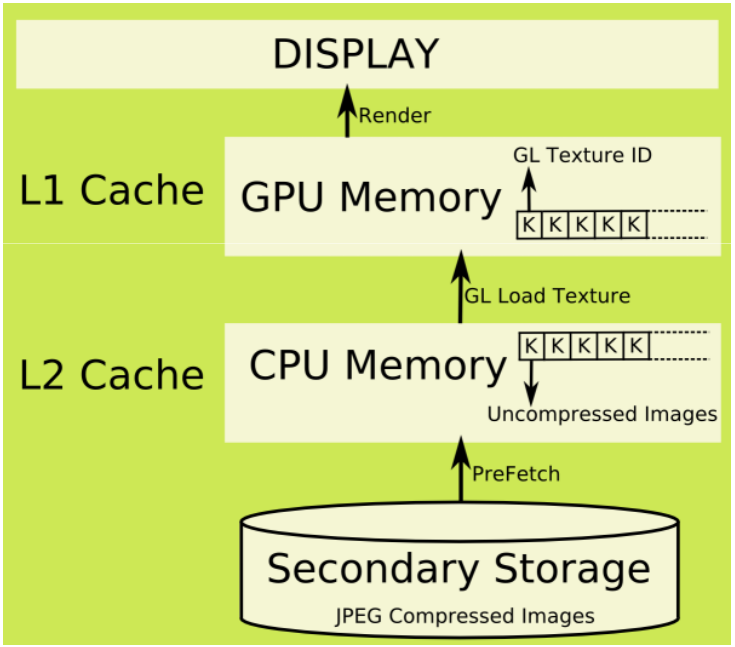
- Intuitive interface
- Acquisition
  - Initialize light-field parameters.
  - Obtain the camera positions.
  - Capture the images.
  - Store the data images.
- Rendering





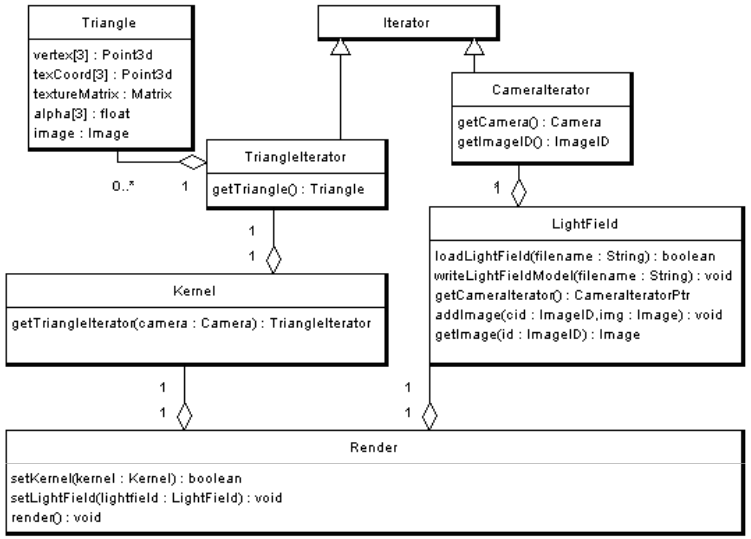
API Design: Rendering

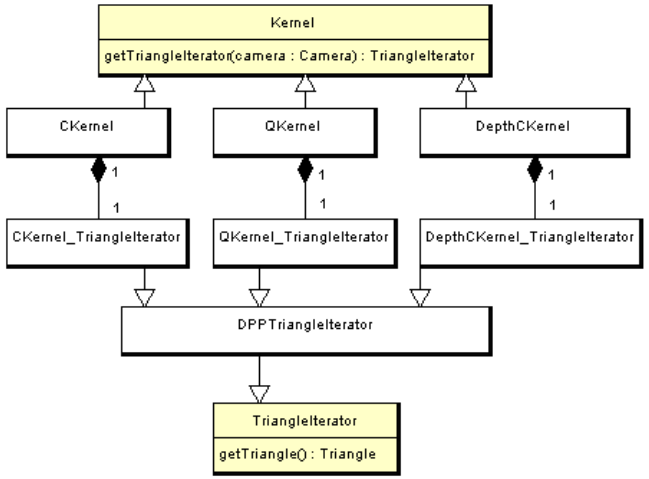
- Uses multiple threads.
- Out-of-core storage techniques.
  - Only new images need to be loaded from disk.
- Two-level cache architecture.
  - Better capacity/response time ratio.



Implementation

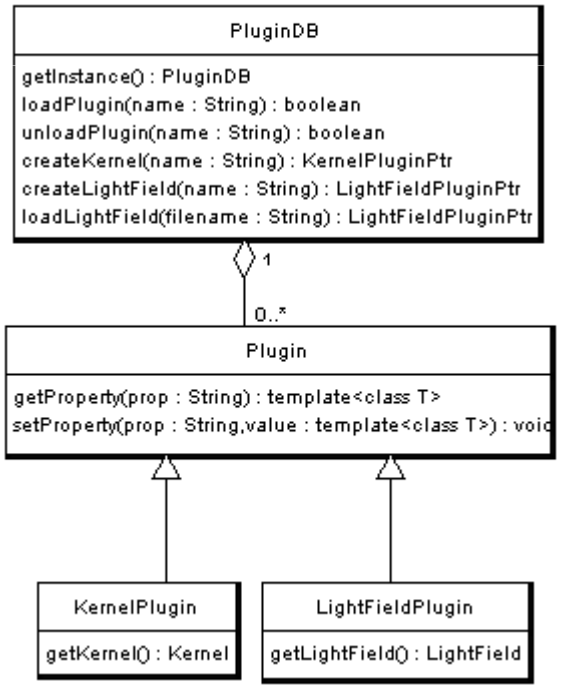
- C/C++
- Interfaces
  - Light-Field Interface
    - Load, save and generate light-field representations.
  - Kernel Interface
    - Reconstruction kernels for rendering algorithms.





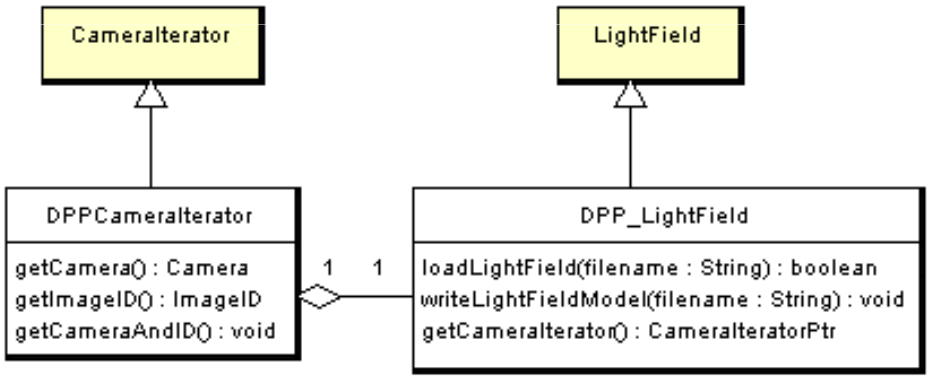
Implementation

- Plugin Architecture.
  - Multiple light-field parameterizations.
- Two kinds of plugins
  - LightField plugins.
    - Planar anisotropic.
    - Spherical isotropic.
    - Unstructured.
  - Kernel plugins.
    - Constant.
    - Linear.



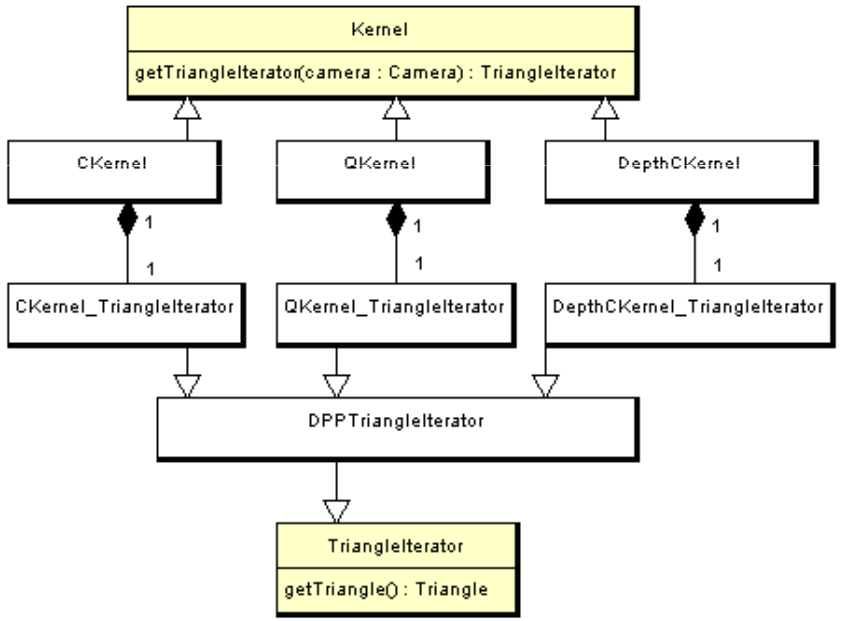
Adding a new LF representation

- Requires adding
  - New class inheriting from the LightField interface
  - Important methods
    - loadLightField
    - writeLightFieldModel
    - getCameraIterator



Adding a new LF representation

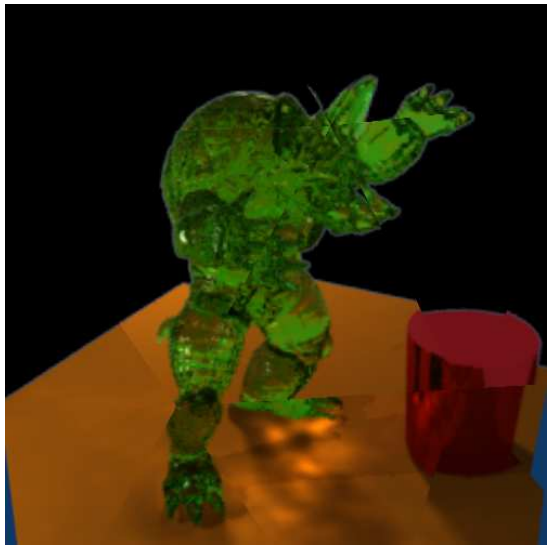
- Write a new CameraIterator.
  - To acquire light fields.
- Implement a new rendering algorithm.
  - Inherit from the Kernel interface.



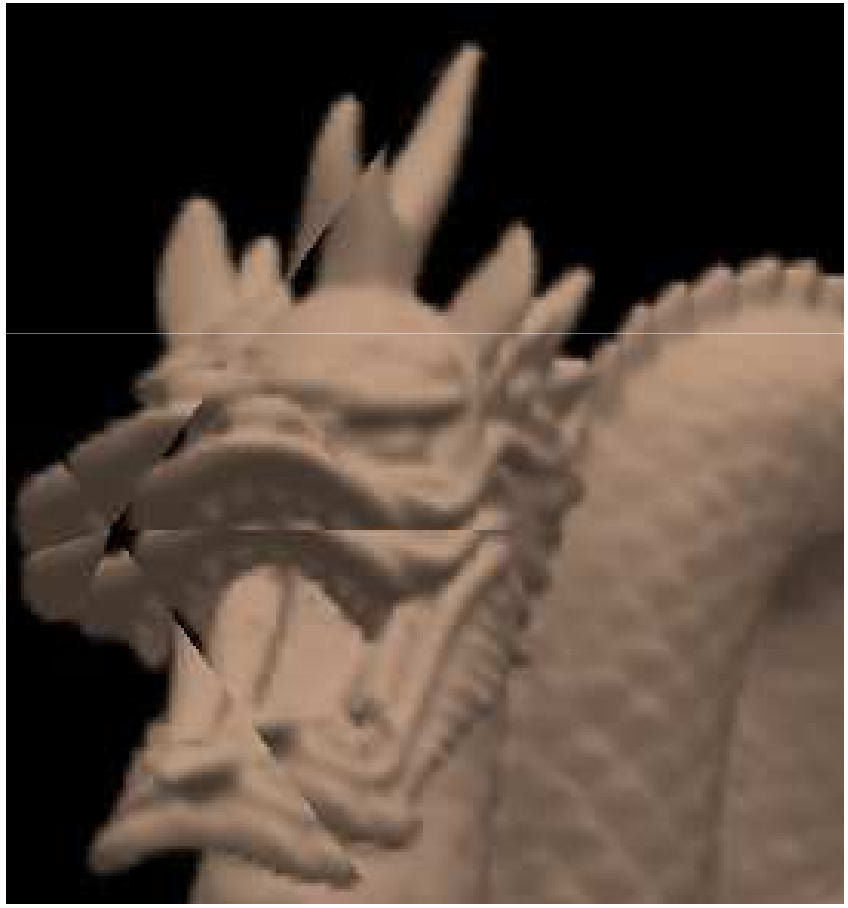
Results

- API supports multiple light fields.
  - Planar and Spherical implemented.
  - Unstructured: in the works.
- Different rendering algs implemented.
  - With constant reconstruction.
  - With linear reconstruction.
  - With depth correction.
  - With integrated geometry.
- Two-level cache management implementation.

Results



Results



Conclusions and Future Work

- Light-field modeling and rendering in a generalized way.
  - Quick development of novel techniques.
- Flexibility and portability.
- Handles
  - Spherical light fields.
  - Planar light fields.

http://www.sig.upv.es/ALF/papers/wscg2008

Questions?

A Generalized Light-Field API and Management System.

Joan Blasco, Miguel Escrivà, Paco Abad, Ricardo Quirós,

Emilio Camahort, and Roberto Vivó

Computer Science Department.

Polytechnic University of Valencia.