Multiple Textures Stitching and Blending on 3D Objects

C. Rocchini, P. Cignoni,
C. Montani, R. Scopigno

Istituto Elaborazione dell'Informazione
Italian National Research Council (C.N.R.)
Pisa, Italy

Presentation Overview

- *The problem* [image-based modeling]
  - given a 3D mesh representing a real objet (e.g. range scanning),
  - how **pictorial detail** can be acquired and mapped on the geometry
  - requiring no special hw and producing standard output
- State of the Art
- Our solution
- Some results and conclusion

State of the Art

Pictorial detail acquisition ==> Image-based Modeling

Acquisition

- ✧ use rgb-enabled 3D scanners (syncro geom & rgb)
- ✧ use special hw to determine the respective locations of the object and the video acquisition device [Sato etal Sig97]
- ✧ register images onto geometry using markers (auto) or maching points (user-assisted)

Mapping

problems: multiple images; how to merge, to map to geometry and to reduce color or pictorial discontinuity

- ✧ divide the surface in disk-homeomorphic regions, for each region resample a texture patch by blending all visible images [Marschner'98]

Our approach

We propose a **sw-only** solution:

- compatible with any geometry acquisition technology (range scanning, medical CT, image-bas. modelling, etc...)
- pitorial detail acquired with a low cost standard device (digital still camera or videocamera)
- output in standard formats (textured mesh -- OpenGL, VRML, Java3D)

Features:

- very high quality pictorial detail (preserved pictorial continuity)
- smooth transition between mapped textures
- works on 3D models with complex topology
- efficient process, limited user intervention (registration phase)

Texture blending is not a easy task

shaded mesh

naive mapping
(Cyberware textures)

a better result

Detail Acquisition & Registration

- Acquisition:
  - use a standard digital still camera;
  - shoot multiple images from different directions, covering all object surface.
- Registration (& camera calibration):

  we developed a simple tool
  - user selects a [small] set of corresponding points on the image and the mesh;
  - the tool computes the camera calibration parameters and view specs

    [Tsai ’87]

Texture ***Stitching&Blending*** Steps

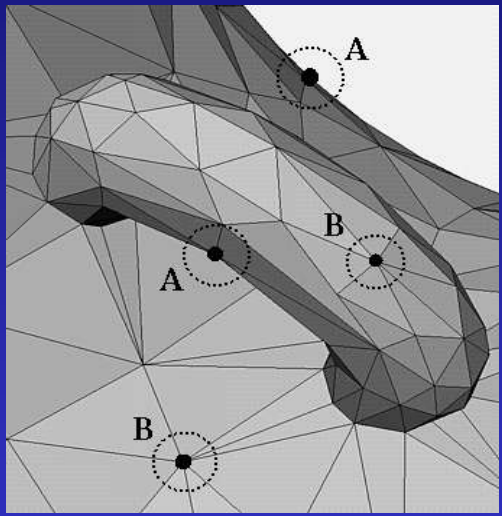
1) *Vertex - to - Image* Binding
2) Patch Growing
3) Patch Boundary Smoothing
4) Texture Patches Packing

Vertex-to-Image Binding (Phase1)

Goal: assign to each Vertex ***v*** a set of valid images and a target image

- Selection criteria for the ***valid image set***:
  - *v* must be visible in the image (this test requires ray-casting for topologically complex surfaces);
  - *v* must **not** be a *silhouette* vertex.
- Selection criteria for the ***target image***:
  - the most orthogonal (view direction) to the surface *M* in *v* among all valid images.

(Vertices not visible in any images are detected, to help the user to produce additional images).

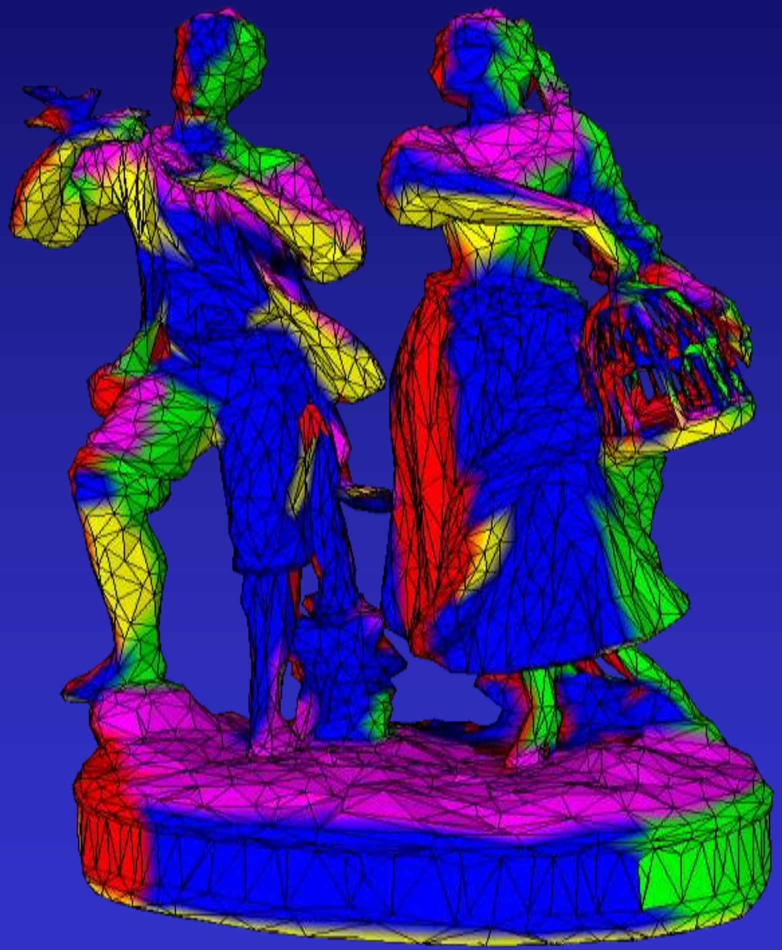


A) Silhouette

B) Non Silhouette

...Vertex-to-Image Binding

- The “Lovers” Mesh: an example of vertex-to-image binding on a complex surface.
- Each color represents a different image.

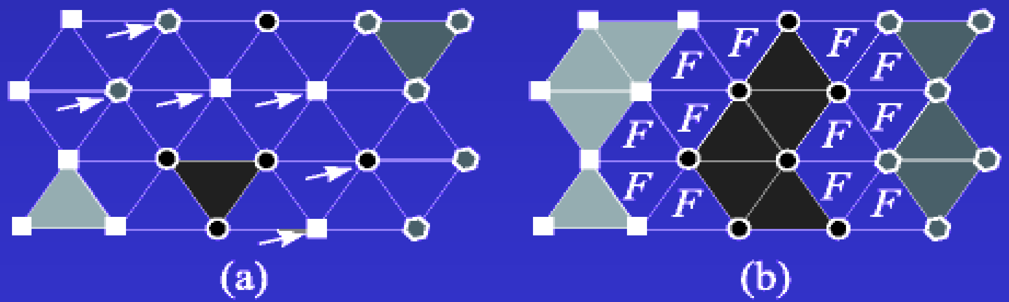


Patch Growing (Phase 2)

- **Goal**: reduce the number of ***frontier*** faces (= whose vertices are associated to 2 or 3 different target images).
- **Motivation**: minimize texture resampling, because we resample the texture image associated to each *frontier* face.
- A greedy iterative algorithm is applied:
  - we change the target face of each vertex (within its valid set) to reduce the local number of frontier faces;
  - a vertex can change multiple time its target face, until we get a minimum.

The set of vertices indicated with the arrows change their target face.

The face marked with "F" are resampled.

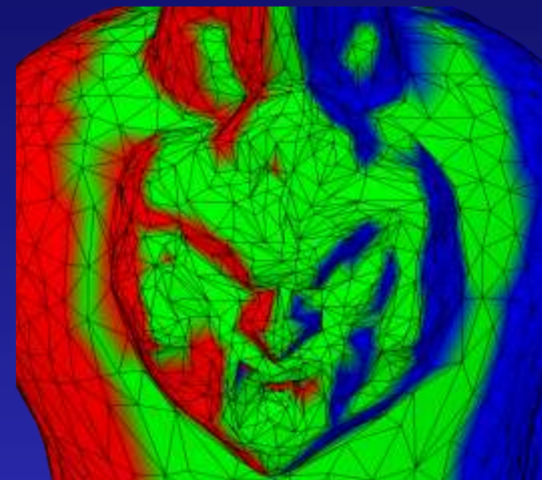


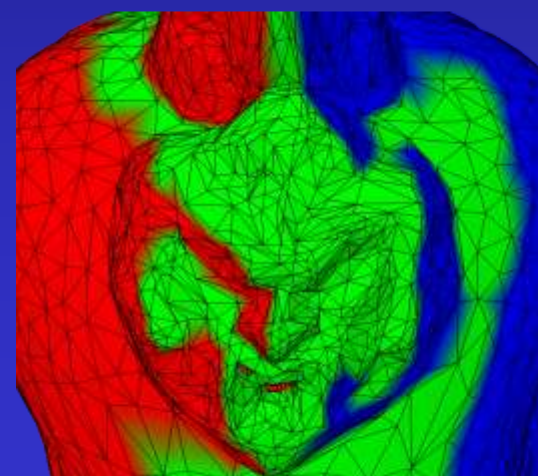
… Patch Growing: an example

Mesh: ***“vase”***

- Total faces (in this case): 10,600
- Frontier faces:
  - ✧ Before PatchGrow: 1,137 (10.7%)
  - ✧ After PatchGrow : 790 (7.4%)

(The three colors correspond to three different target images)





Boundary Smoothing (Phase 3)

Goal: reduce discontinuity on *frontier* faces.

- **Sources of Registration Errors** :
  - imprecise selection of corresponding point pairs;
  - simplified camera model;
  - limited numeric precision in the computations.

- **Solution**: *Local Registration Process*
  - project each pair of corresponding image sections (centered on a frontier vertex ***v***) in the same space;
  - compute a local image registration in 2D and produce new texture coordinates for the given vertex ***v*** ;
  - texture resampling: cross-fade the pair of images using the new texture coordinates.

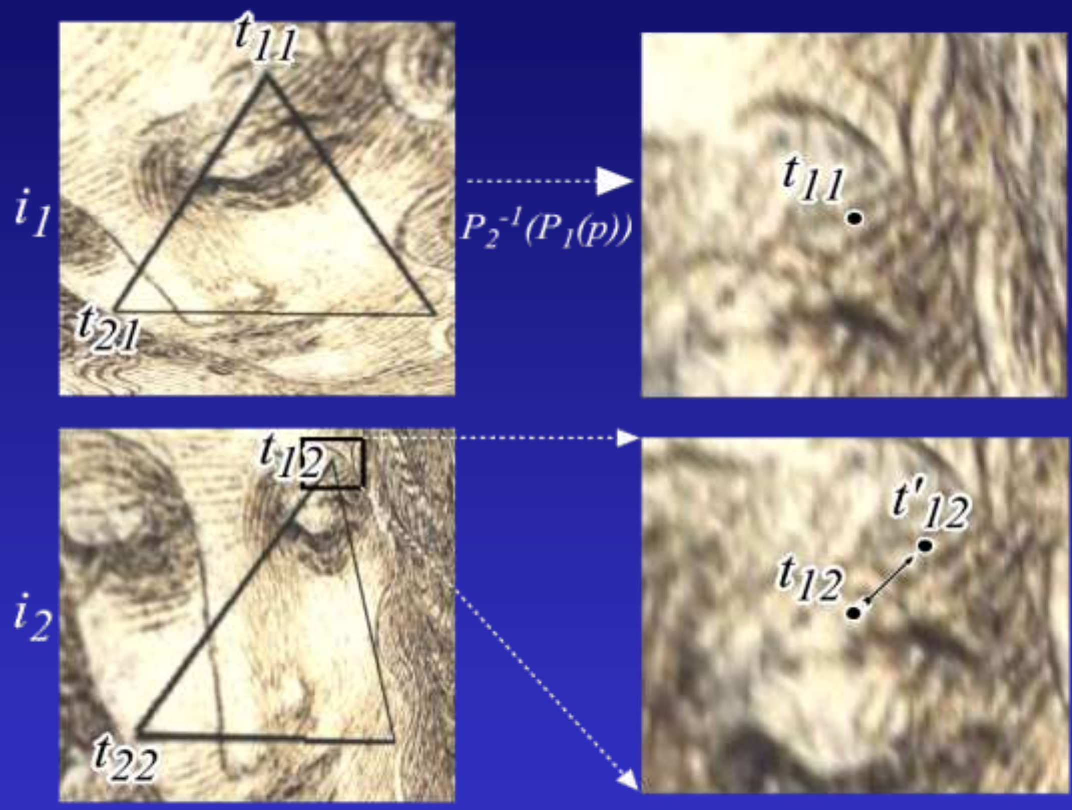
… Bound. Smooth.: Local Registration ...

For each frontier face **f**

for each vertex v

- given $\boldsymbol{i_1}$, $\boldsymbol{i_2}$ the images associated to face ***f*** (and $\boldsymbol{i_1}$ is the target image of ***v***)
- project $\boldsymbol{i_1}$ in the same space of $\boldsymbol{i_2}$ via $\boldsymbol{P_2^{-1}(P_1(t))}$;
- apply a 2D **local registration** algorithm on the two image sections centered on v (i.e. maximize cross-correlation);
- compute the new texture coordinates of $\boldsymbol{v_1}$ in $\boldsymbol{i_2}$;

Scan-convert face ***f*** and resample the corresponding texture triangle (blending the 2 or 3 target images)

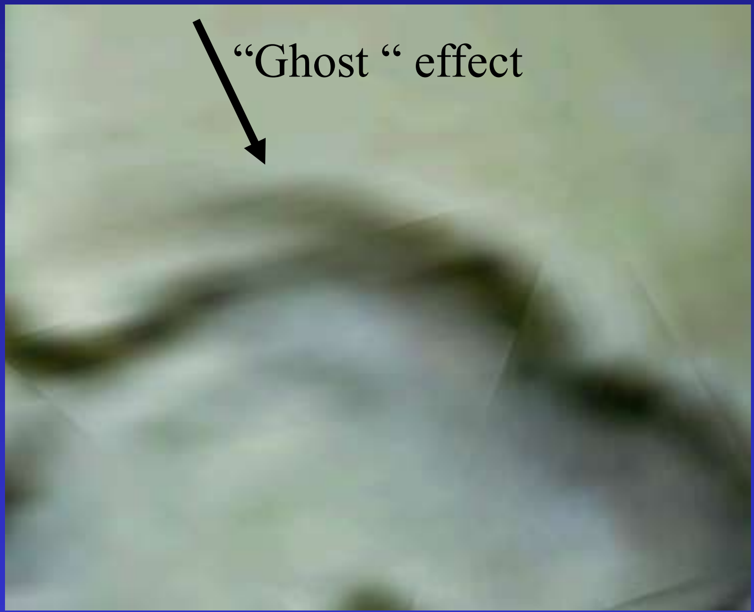


Note: The projection coordinates of the target image are always fixed.

… Bound. Smooth.: Local Registration Sample (1) ...

An example of the improvement due to local registration

Local Registration OFF



Local Registration ON

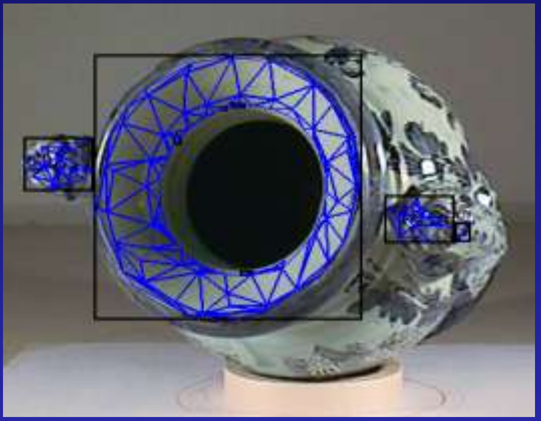
… Bound. Smooth.: Local Registration Sample (2)

Two target images are used

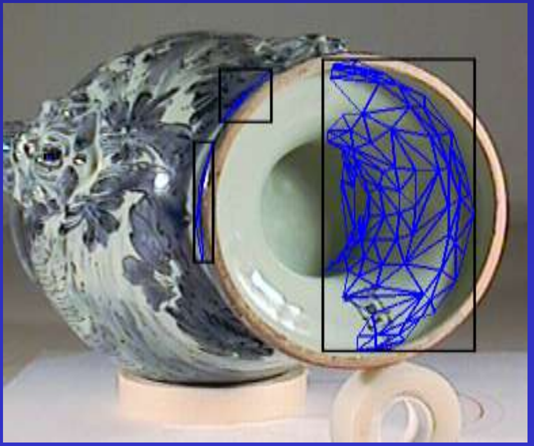
the stripe of frontier faces is marked by the thick white polyline

Texture Packing (Phase 4)

Goal: build a single standard texture image for the whole mesh, render on standard graphics systems (VRML, OpenGL, Java3D)



- Construction steps:
  - **extract** from each target image the minimal subset of rectangular fragments that cover the mapped area
  - **pack** all rectangular fragments and resampled triangular texture patches using a *cutting-stock* algorithm



… A Texture Packing Sample

The final "Vase" Texture: 1024x450 pixels

Some Results (1/2)

Mesh: ***vase***

- size: ~40 cm
- faces: 20,000
- photos: 8
- cpu time: 191 sec.*

Notes: very complex pictorial data

* On SGI O2 R5000

Original Object

Synthetic Model

… Some Results (1/2)

- Show ***Vase*** Mesh demo...

Some Results (2/2)

Mesh: ***"Lovers"***

- size: ~25 cm
- faces: 10,000
- photos: 14
- cpu time: 132 sec.*

Notes: very complex geometry

Original Object

Synthetic Model

* On SGI O2 R5000

Conclusions

- A system for the semi-automatic acquisition and mapping of pictorial detail for 3D objects.

- *Features*:
  - requires only cheap hardware (a simple digital photocamera);
  - limited user intervention (global registration);
  - manages objects with complex geometry;
  - capable of acquiring and mapping very complex pictorial detail;
  - high quality multiple pictures blending, with limited texture resampling (=> texture quality is preserved);
  - output in standard format (i.e. OpenGL, VRML), does **not** require a special-purpose Viewer.

Thanks

Holly Rushmeier and Fausto Bernardini, IBM Yorktown

How to contact us

http://vcg.iei.pi.cnr.it

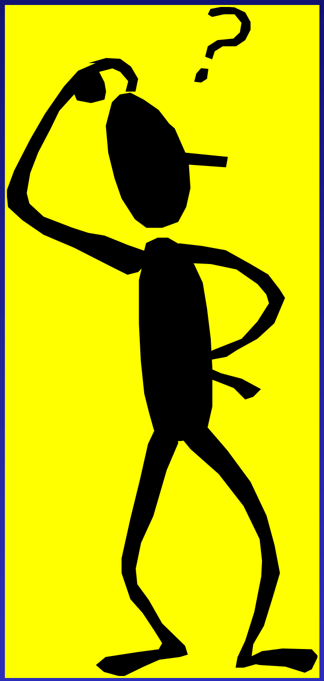
Do not miss...

Eurographics ‘99 Conference

“Bringing to new life our Cultural Heritage”

Milano (Italy), Sept. 7-11, 1999

Questions?



- **Web: http://vcg.iei.pi.cnr.it**

Texture de-shading

- Removing lighting effects (highlight, shading, shadows) is crucial
- We shot **6** images for each view, with different lighting conditions (lights driven by sw)
- For each set of images
  - remove highlights (intensity peaks)
  - remove shadows (low intensity pixel values)
  - de-shading using 3D geometry (simple Lambertian model) and the remaining pixel values (min 3)

Will be described in detail in a forthcoming extended paper...