Real-Time Virtual Viewpoint Generation on the GPU for Scene Navigation

May 31st 2010

Presenter: Robert Laganière

CoAuthor: Shanat Kolhatkar

Contributions

- Our work brings forth 3 main contributions:
  - The calculation of new viewpoints in real-time using a view morphing algorithm
  - The ability to navigate the environment in real-time through the use of GPU programming
  - Enhancing the optical flow to reduce the number of noticeable artifacts

Previous Works

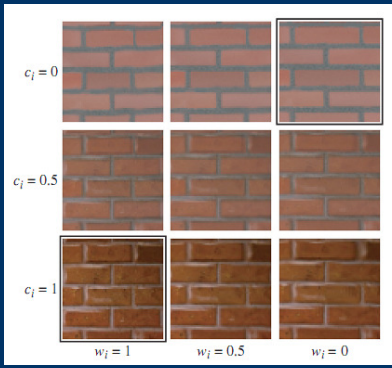
- Optical Flow calculation:
- Ogale and Aloimonos Algorithm
  - Handles occlusions
  - Contrast invariant matching function
  - Indentification of slanted surfaces
- Implementation available for free at:
  http://www.cs.umd.edu/ogale/download/code.html

Previous Works

- Texture Morphing [2]
  - Linear color interpolation and motion compensation to generate the composite texture
  - Doesn't depending much on user input
  - Works with a wide variety of textures
  - Produces high quality results
  - Needs to have features of similar sizes in both origin and target textures.
  - Cannot morph smoothly between highly different textures.

Previous Works

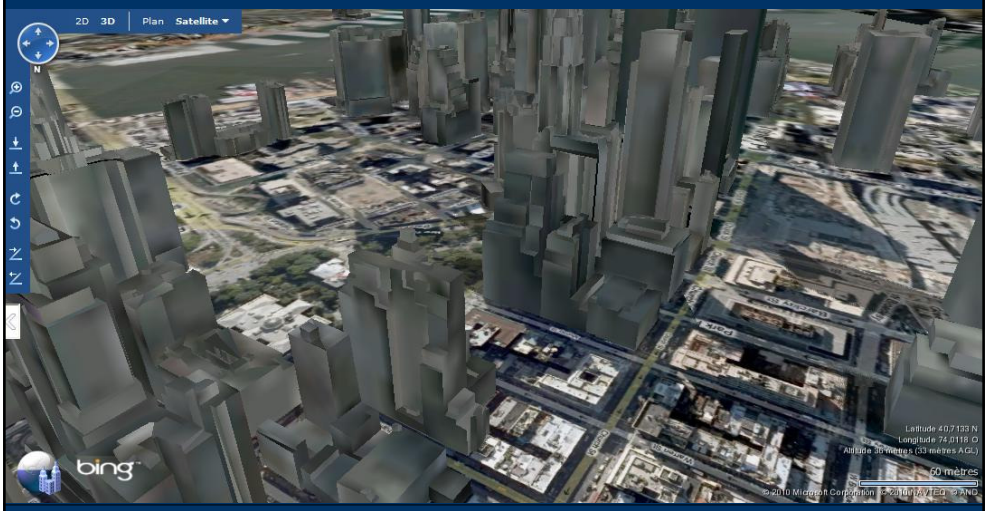
- $$f(x,y) = \sum_{i=0}^{n-1} c_i I_i \left((x,y) + \sum_{j \neq i} (w_j W_{ij}(x,y)) \right)^{-1}$$
- Termes:
  - ci, wj : Color and distance weights
  - Ii: Color map of texture i
  - Wij: Displacement map from texture i to texture j



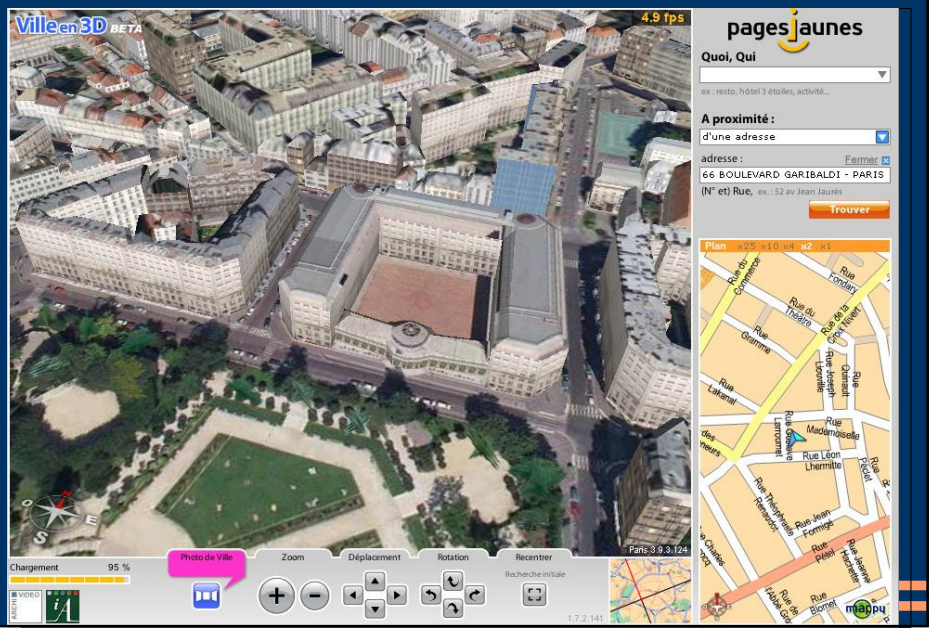
Virtual Navigation Software

- Using panoramas:
  - Google Street View
  - EveryScape
- Using 3D reconstruction
  - Bing
  - Google Earth Feature
  - VillesEn3D

Bing

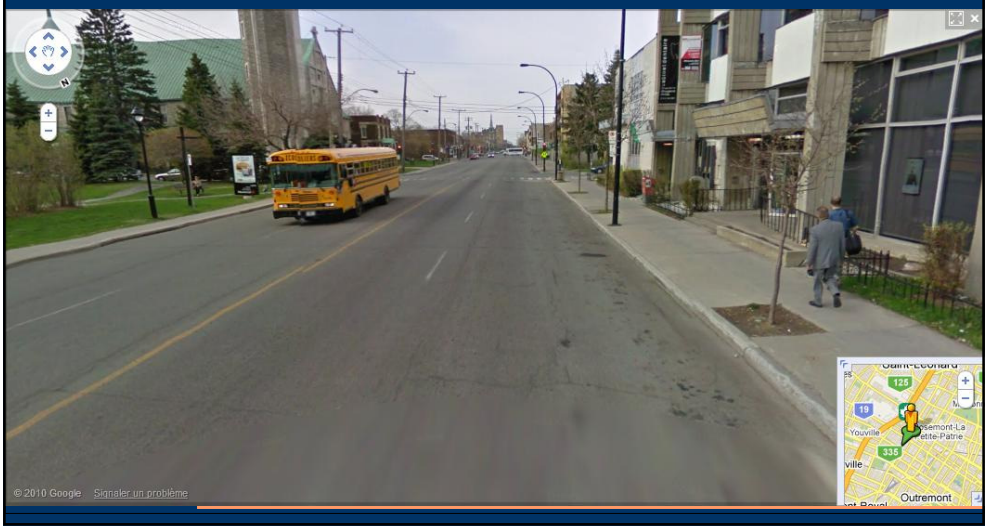


VillesEn3D



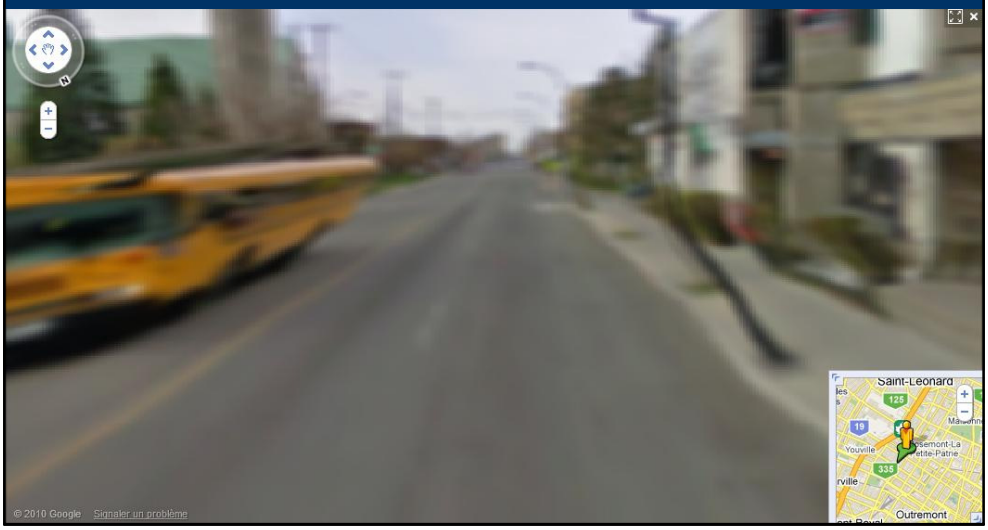
Google Street View

A Captured Panorama



Google Street View

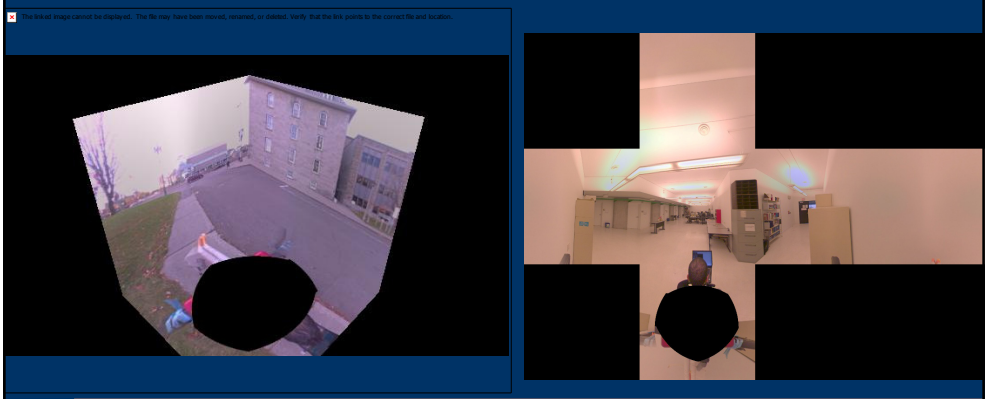
A Transition Image example



Part 1: Enhancing the Optical Flow

Part 1: The Extended Cube

- An example of the previously used cubic panorama representation

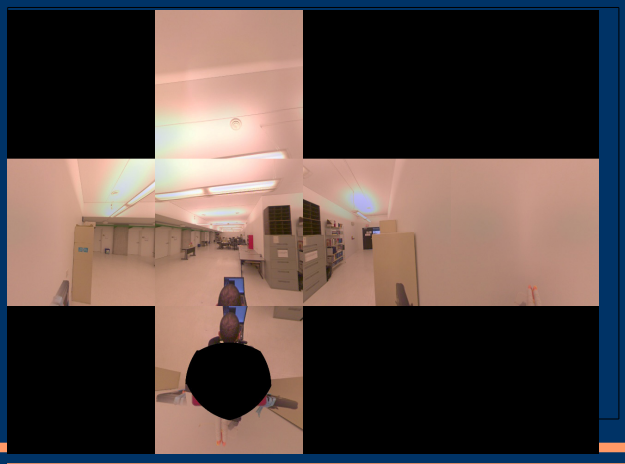


Part 1: The Extended Cube

- Description of the extended cube
  - Each face is extended a certain number of pixels to each side during generation
  - Goal: Correctly estimate the displacement of a visual point moving from one face to another. (small displacements)

Part 1: The Extended Cube

- Extended Cube example

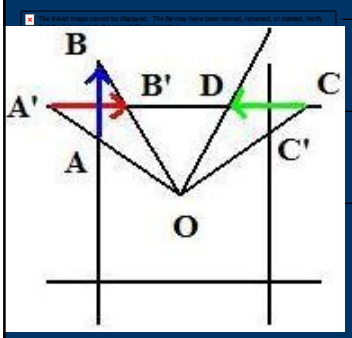


Part 1: The Extended Cube

- 3D View of the Extended Cube

Part 1: The Extended Cube

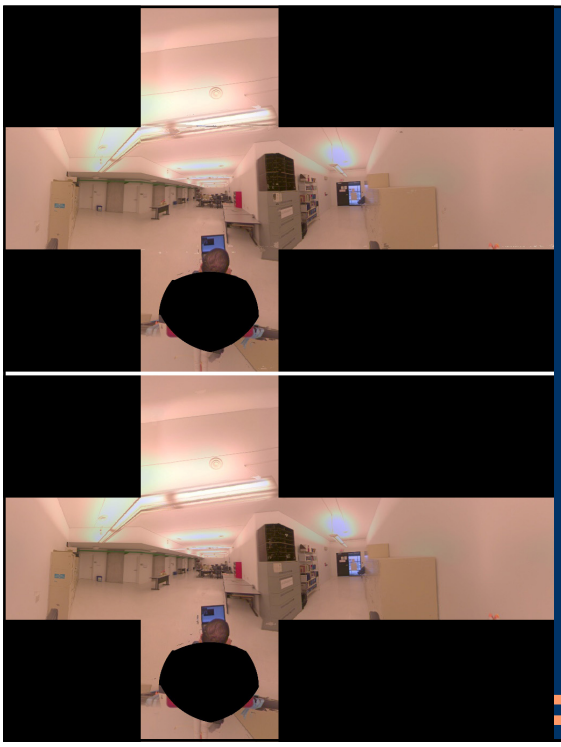
- Boundary Handling



Some displacement vectors are computed twice because they appear on each extended portion of two adjacent faces

We check which of these two solutions gives the best result by comparing the color neighborhood in both images according to the L2-norm

The one with the greatest distance is then replaced by the other one

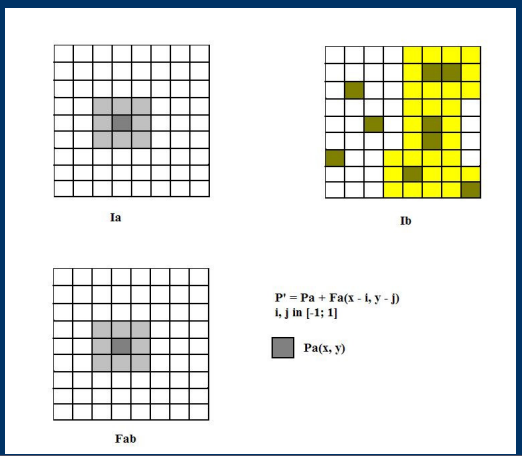


Interpolated image using a cube

Interpolated image using our extended cube

Part 1: Smoothing The Flow Field

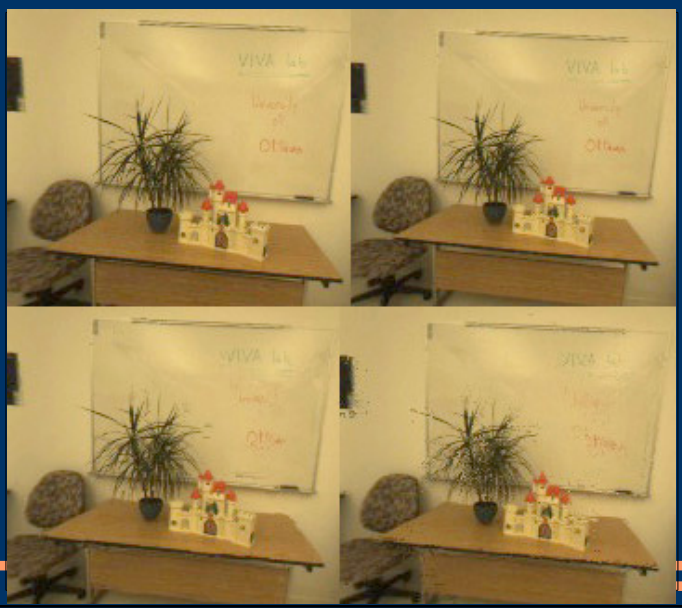
- We assume that the image is static.
  - neighboring flow vectors should have a similar orientation



Part 1: Smoothing The Flow Field

Original

Destination



Smoothed

Uncorrected

Part 2: View Interpolation

Part 2: View Interpolation

- Calculate each extended face of the cube independently from the others
- We use the previously calculated dense flow field to interpolate the position of the pixels
- We want the color and the motion to be interpolated jointly thus, our interpolation is written as:

$$\hat{I}(x,y) = (1-c)\,I_0(wW_{01}(x,y)^{-1}) + cI_1((1-w)W_{10}(x,y)^{-1})$$

Part 2: View Interpolation

$$\hat{I}(x,y) = (1-c)\,I_0(wW_{01}(x,y)^{-1}) + cI_1((1-w)W_{10}(x,y)^{-1})$$

- $\hat{I}$ is the transition image
- I_0 and I_1 are the origin and destination images respectively
- W_{ij} is the dense displacement field from image I_i to I_j. The optical flow algorithm that we use makes the assumption that $W_{01}(x,y) = -W_{10}(x,y)$
- c and w are the weights applied to the color and displacement respectively, and depend on our position between both images. In our experiments, we set $w = c$, because we want the color and the geometry of the scene to be interpolated jointly.

Part 3: Real-Time Navigation

Part 3: Real-Time Navigation

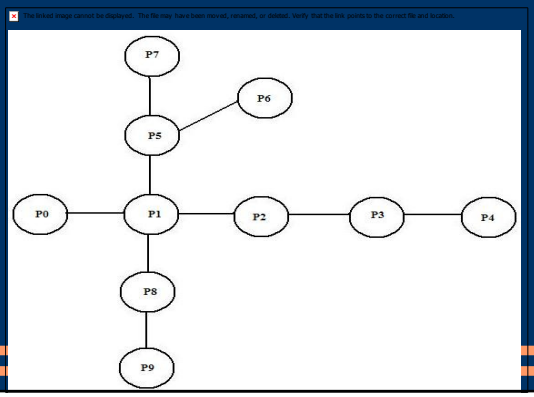
- Four Components are necessary to achieve Interactive Navigation
  - Pre-calculation of the flow
  - Multi-Threading
  - Buffering
  - GPU Implementation of the interpolation

Part 3: Multi-Threading

- Use 3 Threads:
  - One thread to handle the graphics calls (in our architecture, the OpenGL calls)
  - One to load the data from the hard drive
  - One thread to calculate the interpolated images, which is actually run on the GPU

Part 3: Buffering

- This graph is an example of a possible panorama setting. Let us suppose that our buffer size is 7. If the user is currently at the viewpoint P1, then the buffer is filled with the panoramas and displacement fields of the following viewpoints: P1, P0, P2, P5, P8, P3, P6.



Part 3: GPU Implementation

- Each pixel's interpolation is calculated independently from each other
  - we just need the origin and target pixels color values and the flow vectors)
  - This makes it a perfect algorithm for GPU implementation

Part 3: GPU Implementation

- GPU Algorithm

```
uniform sampler2D: OriTex, TarTex, FFX, FFY;
uniform floats: fDispIC, fColorIC, fDispLgthX, fDispLgthY, fTexSz;
TEXCOORD: pixCoord
vec2 iPixCoord = pixCoord * fTextureSize;
vec2 offset = vec2(0.0f, 0.0f);
offset_x =(texture2D(FFX, pixCoord)-0.5);
offset_x *= fDispLgthX;
offset_y =(texture2D(FFY, pixCoord)-0.5);
offset_y *= fDisplacementIntervalLengthY;
vec2 texcoordOrigin = (iPixCoord - fDispIC * offset)/fTexSz;
vec2 texcoordTarget = (iPixCoord + (1.0f - fDispIC) * offset)/fTexSz;
//Interpolate the colors
gl_FragColor = (1.0f - fColorIC) * texture2D(OriTex, texcoordOrigin) +
fColorIC * texture2D(TarTex, texcoordTarget);
```

Part 4: Results

Part 4: Results

- On the CPU, we created 20 transition images in 3 seconds. With our GPU based interpolation scheme, we could generate up to 1000 images in 3 seconds, with identical quality. (We ran our tests on a Pentium M 1.7Ghz with a Radeon Mobility X700)
- The optical flow calculation and correction took up to an hour on a 320x320 image on an Athlon 64 X2 6000+/3GHz. We ran the optical flow calculation with an interval for both x and y of [-40; 40] and we set the size of all the neighborhoods in the correction pass to 11.

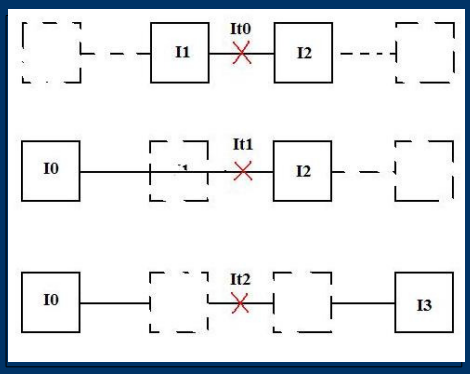
Part 4: Results

- Degradation of results with the increase of the distance between the viewpoints

Step of 1

Step of 2

Step of 3



Step of 1

Step of 2

Step of 3

Part 4: Results

- Smoothing improvement

Uncorrected

Smoothed

Shades of Gray RMS

Origin

Linear

Uncorrected

Smoothed

Ground Truth

Destination

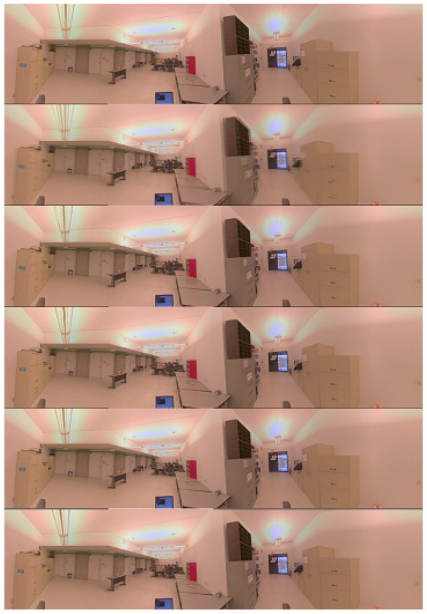


Figure 7.2: Going from the top to bottom image, we have: the origin image; the linearly interpolated image; the uncorrected flow interpolated image; the interpolated image using a smoothed flow; the real image captured at the interpolated location (ground truth); and the destination image. All interpolation uses the best matching parameter $c = 0.45$.

Shades of Gray RMS

Origin Destination

Linear

Uncorrected Smoothed

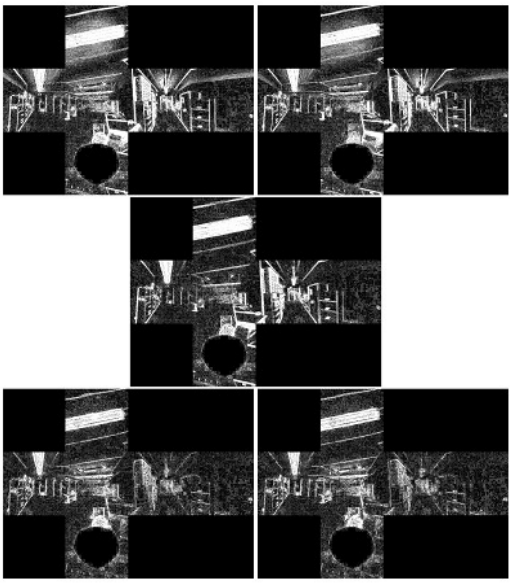


Figure 7.3: Going from top to bottom image (and left to right), we have the comparison images of the real panorama captured at interpolation location with: the origin image; the target image; the linearly interpolated image; the interpolated image using uncorrected flow; and finally with the interpolated image using the smoothed flow. All interpolated images use the best matching parameter $c = 0.45$. This is the same sequence as the one in Figure 7.2

RMS Values

RMS Values when comparing the ground truth with the mentioned image				
Origin	Destination	Linear	Uncorrected	Smoothed
18.4965	18.709	14.7515	9.67674	9.14825
25.5349	25.6267	20.6691	13.9703	13.6975
29.5894	31.5904	25.2227	15.8723	14.6142
27.7342	27.4573	22.0606	12.2229	11.2739
22.61	21.49	17.55	14.02	14.05

Table 7.1: Table showing some of our RMS results comparing a ground truth image with: the origin image; the target image; the linearly interpolated image; the interpolated image using uncorrected flow; and finally the interpolated image using the smoothed flow

Conclusion

- New way of interpolating between pairs of panoramas in real-time using the GPU,
- Navigate inside a scene and achieve a high degree of realism.
- Main contribution:
  - interpolation of intermediate viewpoints of a scene in real-time using computed optical flow field.
- Except for a few artifacts, our results are of good quality, as long as the panoramas were taken at reasonable distances from each other.

References

[1] A. S. Ogale and Y. Aloimonos. A roadmap to the integration of early visual modules. Int. Journal of Computer Vision.72:9–25, April 2007.

[2] W. Matusik, M. Zwicker, and F. Durant. Texture design using a simplicial complex of morphable textures. SIGGRAPH, 4:124–125, 2005.