Point Sample Rendering

Efficient Screen Space Approach for HW Accelerated **Surfel Rendering**

VMV03, november 2003

Gaël GUENNEBAUD - Mathias PAULIN
IRIT-CNRS-UPS
TOULOUSE-FRANCE

http://www.irit.fr/recherches/SIRV/VIS/Surfel/index.html

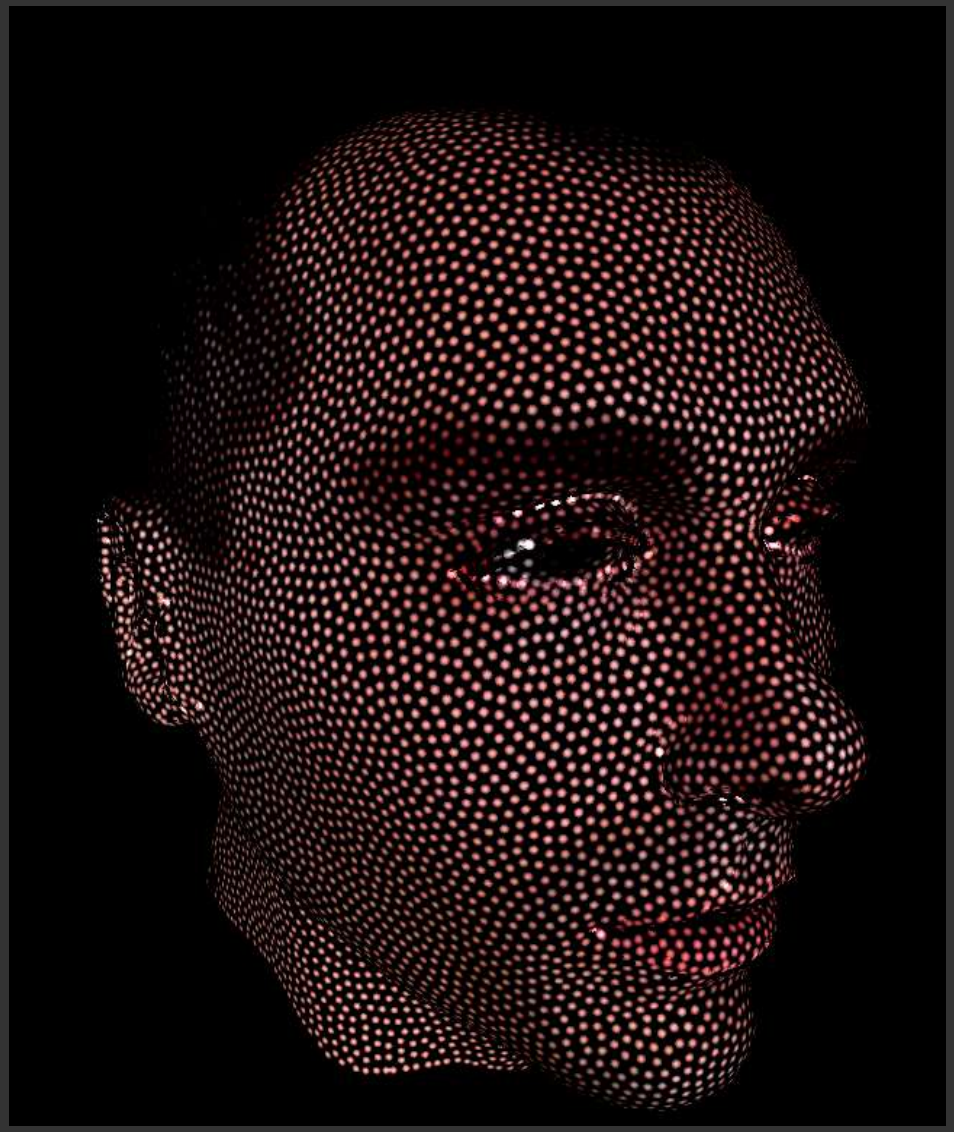
Plan

- Surfel ?
- Related Works
- EWA Splatting background
- Our efficient HW approach
- What about performance ?
- Future work

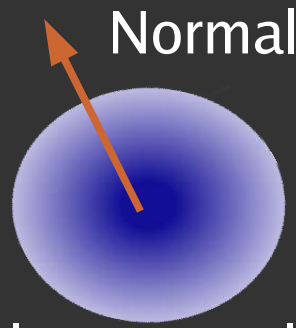
Motivation
of point-based rendering

- High quality rendering of complex objects
  - need very small triangles
  - Building low level of details is very hard
  - ⇒ A simpler, more efficient rendering primitive than triangles ?
- Modern 3D scanning devices acquire huge point clouds (~10 million pts)
  - generating consistent triangle meshes is time consuming and difficult
  - ⇒ A rendering primitive for direct visualization of point clouds ?

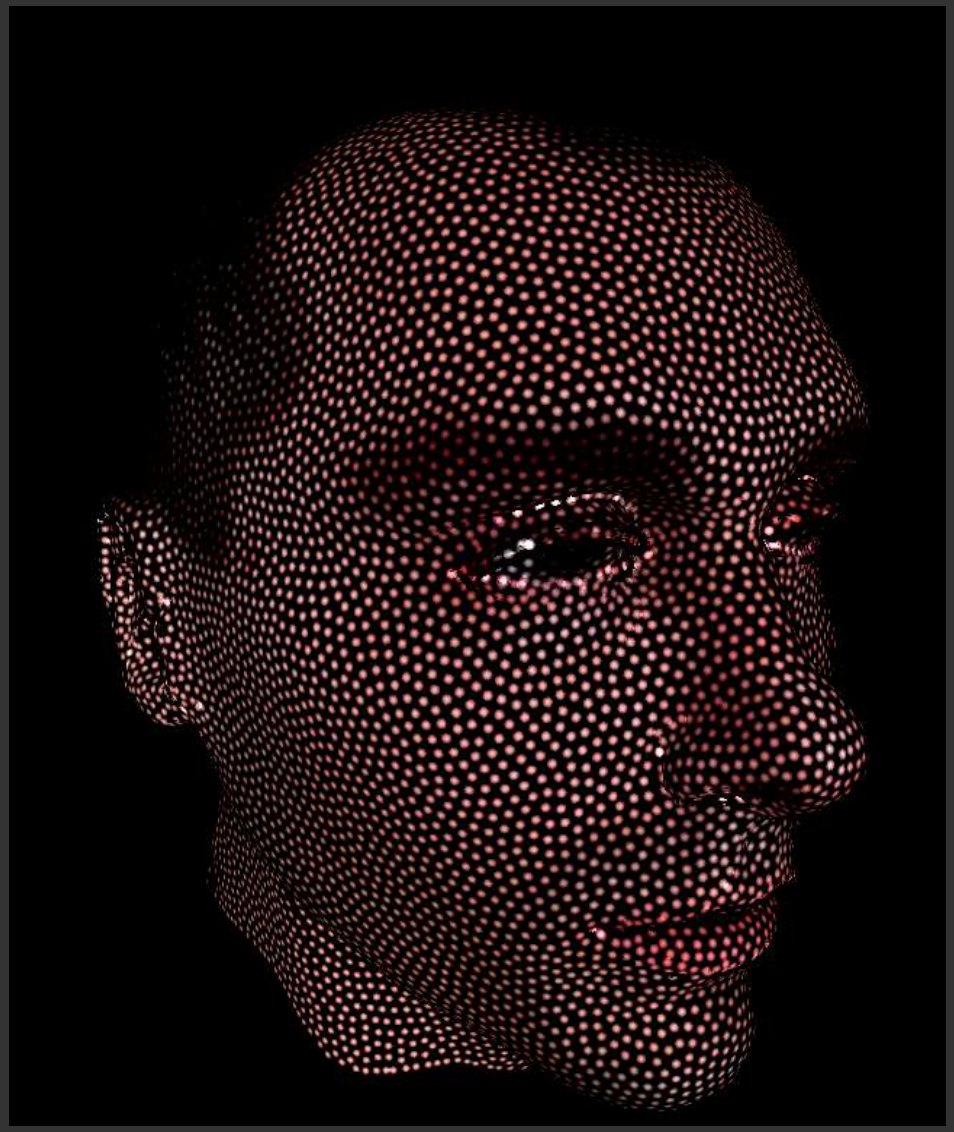
Point-based Surface **Representation**



- The point cloud describes
  - 3D geometry of the surface
  - Surface reflectance properties
- Surface Element (Surfel)

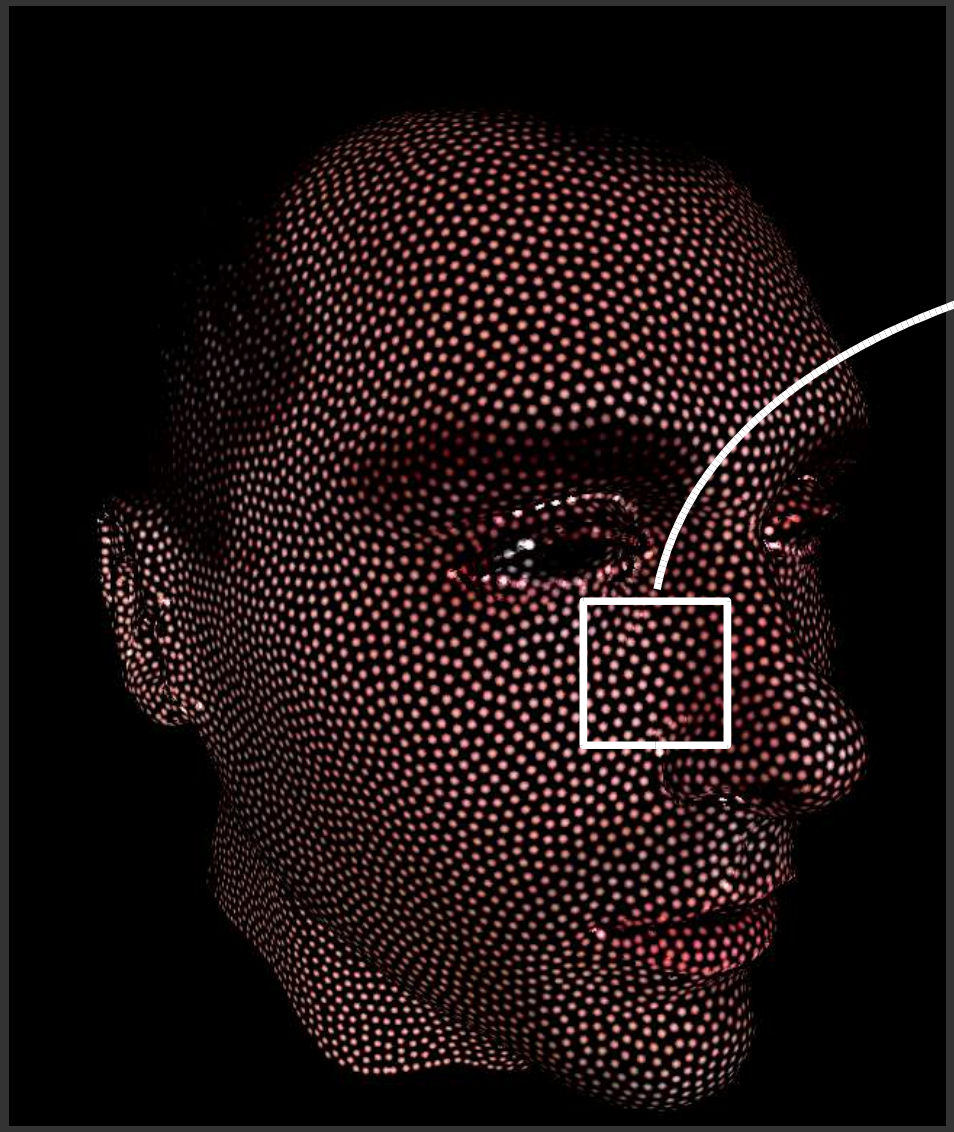


Point-based Surface **Representation**

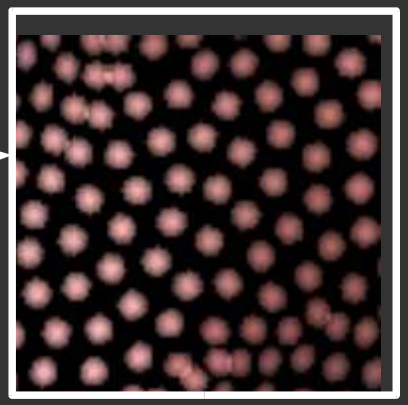


- No additional information
  - No connectivity
  - No texture maps, no normal maps, ...
- Consequently, it's easy to do :
  - Multiresolution
  - Per pixel lighting ~ per surfel lighting
  - Painting
  - ...

What about the **Rendering ?**



- Rendering ...



$=$ holes filling

$\Rightarrow$ Surfels need to interpolate the surface between the points

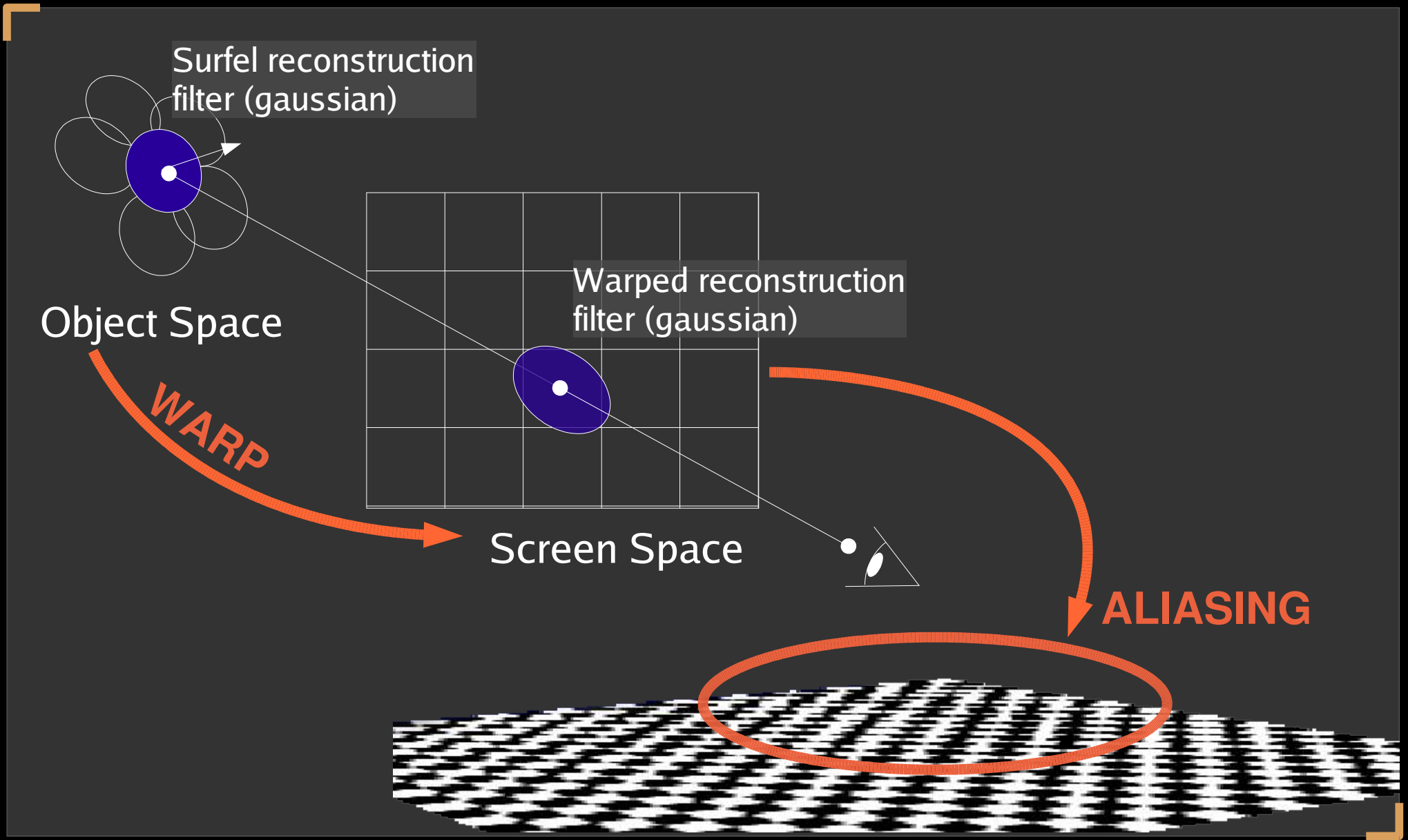
Motivation of this **work**

- Propose a very fast surfel rendering
  - Since GPU performance outpaces CPU, we use the capability of recent GPU
- High quality texture filtering
- Limit memory cost and AGP bandwidth

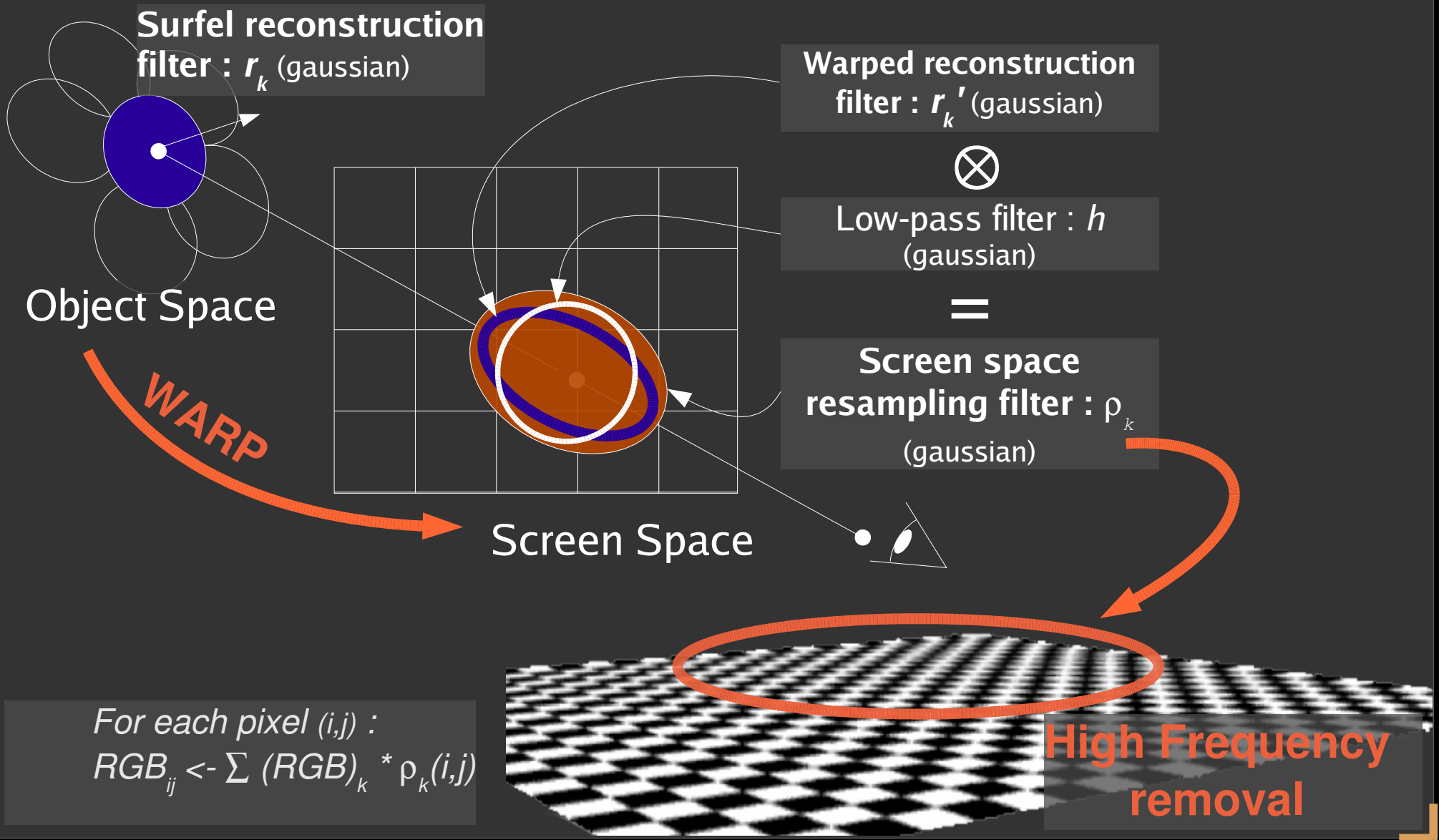
Related Work

	USE HW	Efficiency	Isotropic texture filtering	Texture filtering quality	Memory cost
Point sample rendering [Grossman98]		XX		X	/
Qsplat[RL00] / SPT [DVS03]	X	XXXXX			/
Surfels[PZBG01]		X		XX	/
Surface splatting [ZPBG01]			X	XXX	/
Object-Space EWA splatting [Ren02]	X	XXX	X	XXX	Duplicated data (~x6)
Screen-Space HW Accelerated EWA Slpatting [03]	X	XXXX	X	XXX	/

Splatting


Surfel reconstruction filter (gaussian)
Object Space
Warped reconstruction filter (gaussian)
WARP
Screen Space
ALIASING

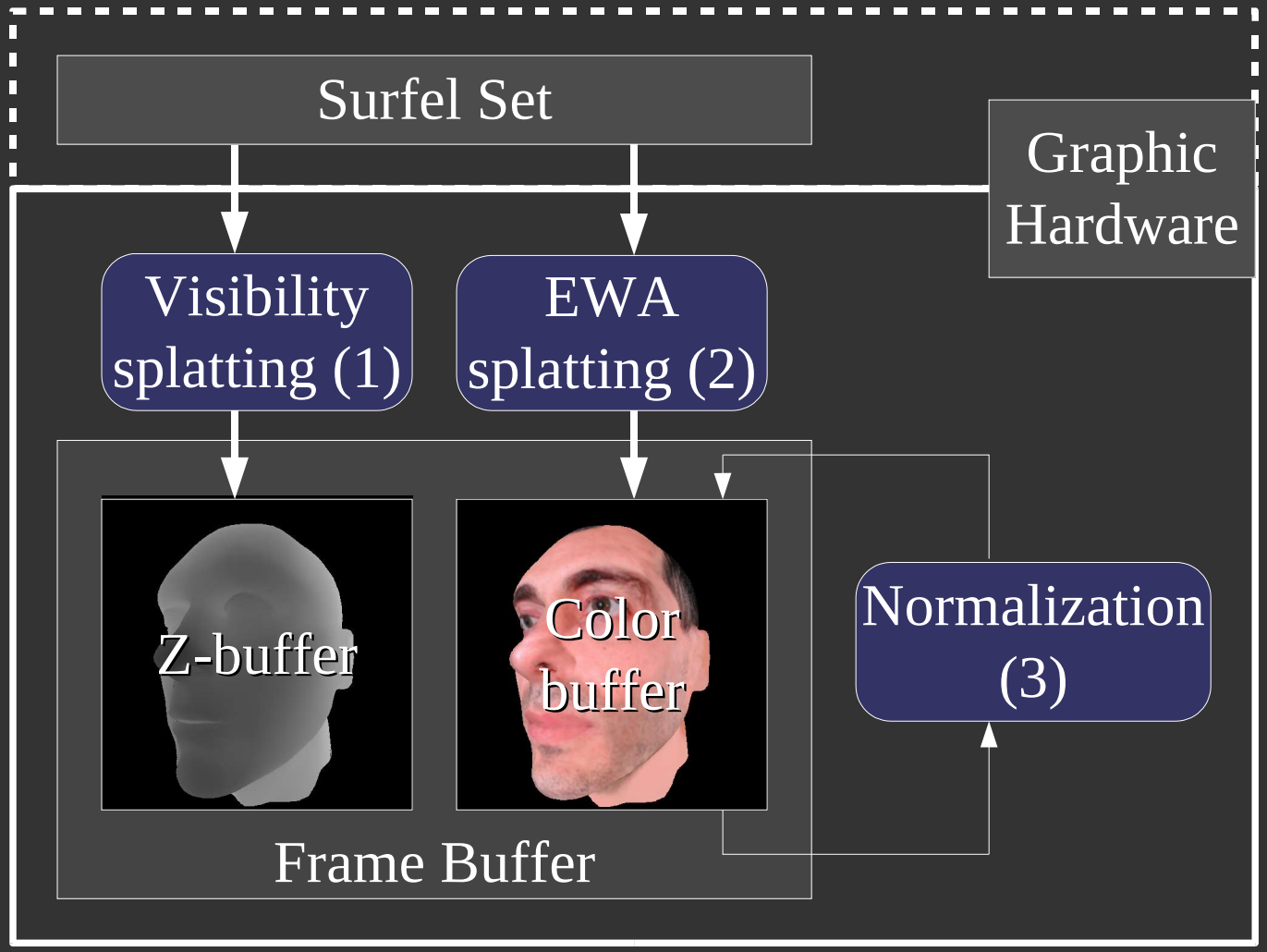
Screen Space EWA Filtering [ZPBG01]



Challenges

- Visibility
  - No hole,hidden surface splats removal
  - Lack of A-buffer support
- EWA resampling filter
  - View dependent
  - Render elliptical Gaussian efficiently
- In order to minimise memory cost and bandwith :
  1 surfel ⇔ 1 vertex ⇔ 1 GL_POINT

Three pass Overview

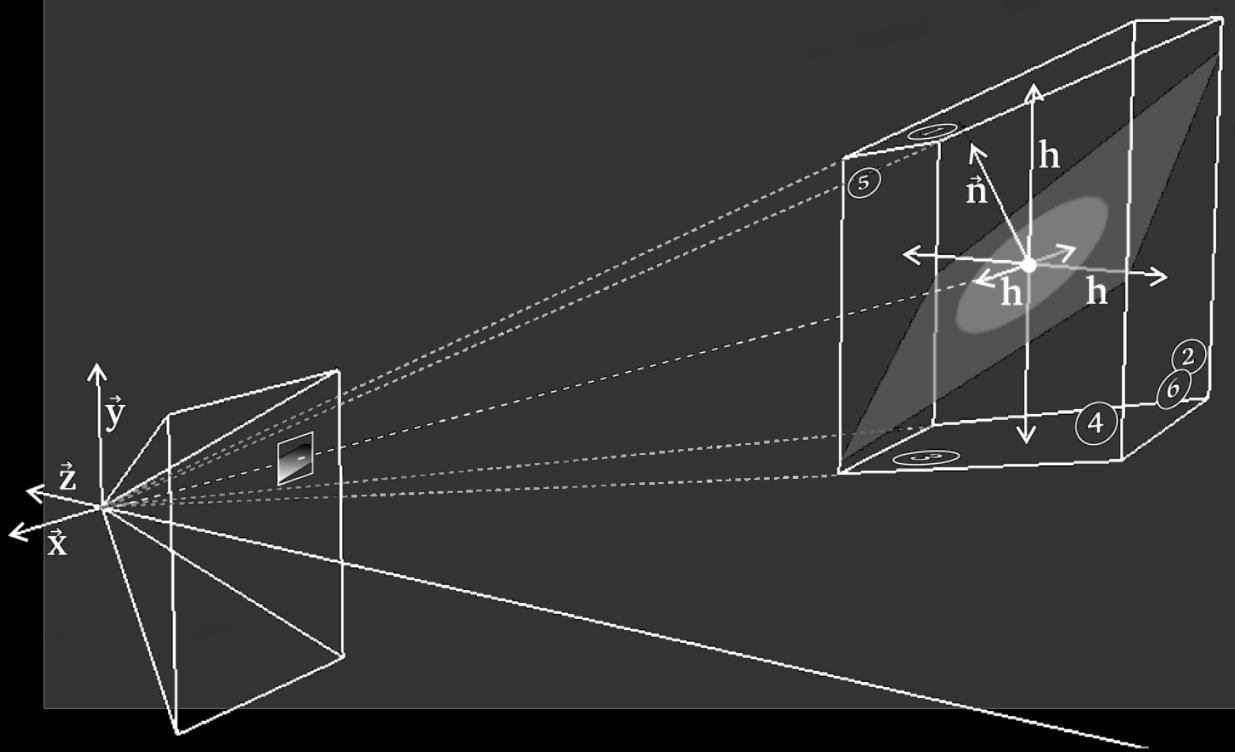


- **Visibility Splatting**
- Generate the depth buffer
- With a small offset
- Without any hole !
- **EWA Splatting**
- Render elliptical gaussian splat
- Use additive alpha blending
- **Normalization**
- RGB = RGB / A

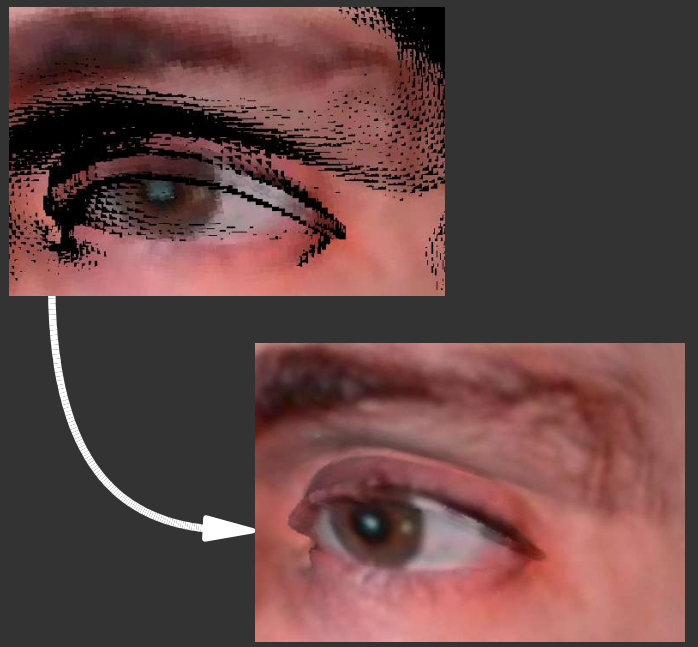
Pass 1 : Visibility Splatting

- How to handle an oriented disk with a simple GL_POINT ?

Surfel = tangent plane clipped by a frustum box



- Per fragment depth correction (via fragment program)



Pass 2 : EWA Filtering

- How to handle a view dependent elliptical disk with GL_POINTS ?
  - Use the GL_POINT as a bounding-square of the resampling kernel (in screen space)
  - Compute the parameters of the resampling filter (a gaussian) in VP (view dependent, per surfel)
  - Compute Mahanalobis distance & exponential in FP

Pass 3 : Normalization

- Irregular sampling & truncation of the Gaussian kernel =>

$$\sum \rho_k(i,j) \neq 1$$

- Use alpha component :

$$A_{ij} \leftarrow \sum \rho_k(i,j)$$

(during the EWA-splatting pass)

- Render the frame buffer in the frame buffer with :

$$(R'G'B')_{ij} = (RGB)_{ij} / A_{ij}$$

Optimizations Issues

- Use a hierarchical data structure
  - View frustum + back-face culling (Speed++)
  - Solve the problem of too many splats on 1 pixel (Quality++)
- Per surfel back-face culling via VP
  - Set *position.w = 0* for back-face surfel
  - Speed++
  - Quality++ (on outline)

HW Implementation

GeForceFX 5800 under Linux

- High quality rendering
- Performance : fast, but the cost of VP and small FP (~5 op.) is very expensive !
  - Max : 60M GL_POINTS per second
  - Normalization cost : 0.12ms !
  - This results are computed without high-level data structures.

FP on/off	Back-Face culling	Head model : ~300k points (512x512)			
		Visibility	EWA splatting	FPS	Surfel / s
	on/off	10ms	14ms	41.6	12.5M
X		28ms	31ms	16.9	5M

Results

Screen Space **VS** Object-Space [ren02]

- Drawbacks :
  - Rasterisation is more expensive
  - Need recent graphic hardware
- Advantages :
  - No data duplication
    memory cost / 6.8
    AGP bandwidth / 3.7
  - No duplicated vertex computation (/4)

  ⇒ Faster : ~3-3.5 X

Future

- Faster implementation
  - Simpler vertex and fragment programs via approximations of the EWA filtering
- Visibility and EWA splatting in one pass ?
  - Via a z-buffer test with tolerance

    if $z<depth_{ij}-e$ then { $C_d<-C_s$; $depth_{ij}<-z$; }

    else if $z<depth_{ij}+e$ then { $C_d<-C_d+C_s$; }
  - But need a R/W access in FP
  - Or support for "BlendingProgram" ?
- Add support for semi-transparent surface

VIDEO


MPlayer
MPLAYERHQ.HU
the movie player for linux - official site

Questions ?