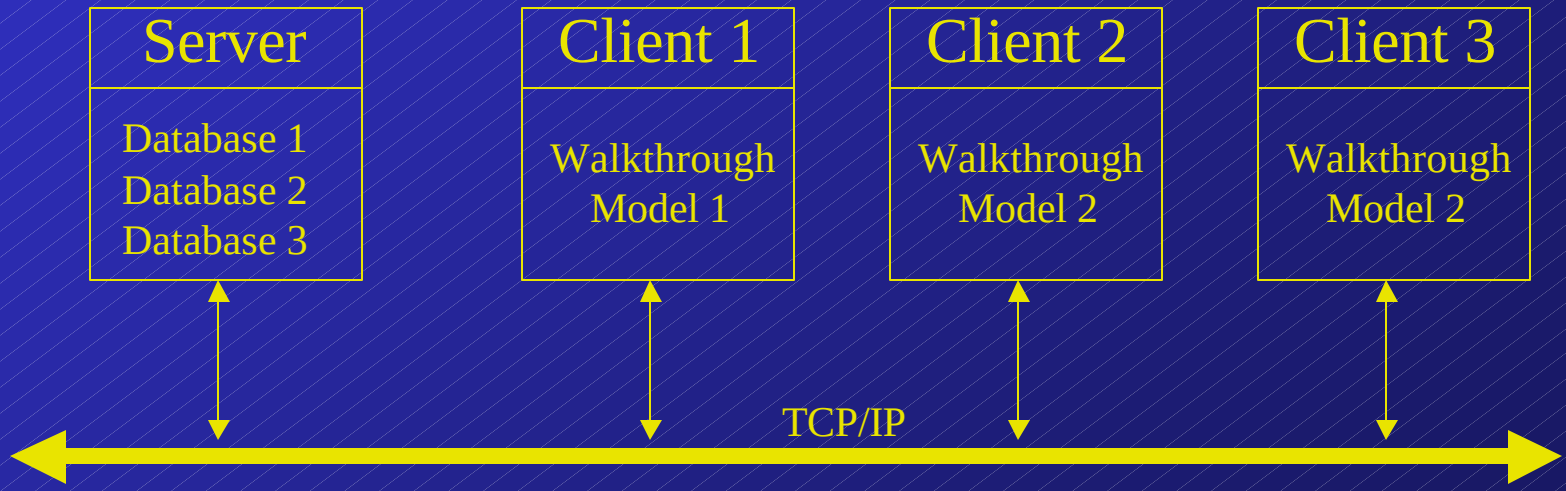
Remote Rendering of Massively Textured 3D Scenes Through Progressive Texture Maps

Jean-Eudes Marvie
INSA of Rennes

Pr Kadi Bouatouch
University of Rennes

Objective

- Real time navigation
- Client-server architecture
- Applied to complex 3D scenes



Constraints

- Complex 3D scenes :
  - Millions of polygons, objects,
  - Textures coordinates, materials, normal vectors,
  - Large set of texture maps
- Real-time navigation : (25fps)
  - Limited by graphics hardware performances,
  - By network bandwidth,
  - By the amount of data to process

Used solutions

- Global optimizations
  - Spatial subdivision of the scenes (set of cells),
  - Visibility preprocessing between the cells,
  - Partial visibility graphs
- Local optimizations
  - Progressive geometry
  - Progressive texture maps
  - Automatic selection and adaptation

Overview

I - Progressive Texture Maps (PTM)

- Texture maps are specific images

II - Average Coverage Hint (ACH)

- A pre-computed selection metric for mipmap levels

III - Automatic adaptation

- Using PTM and ACH to perform real time remote navigation

IV - Results and discussion

- A virtual museum with high resolution texture maps

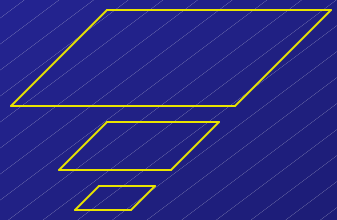
I - Progressive Texture Maps

PTM

A compact file format to encode texture mipmaps

I - Progressive Texture Maps

- JPEG
  - + Hierarchical representation
    - ⇒ Possibility to encode texture mipmaps
  - + Lossless compression
    - ⇒ Average compression equal 2:1
  - – Data redundancy between levels
    - ⇒ Original size increased by 30%



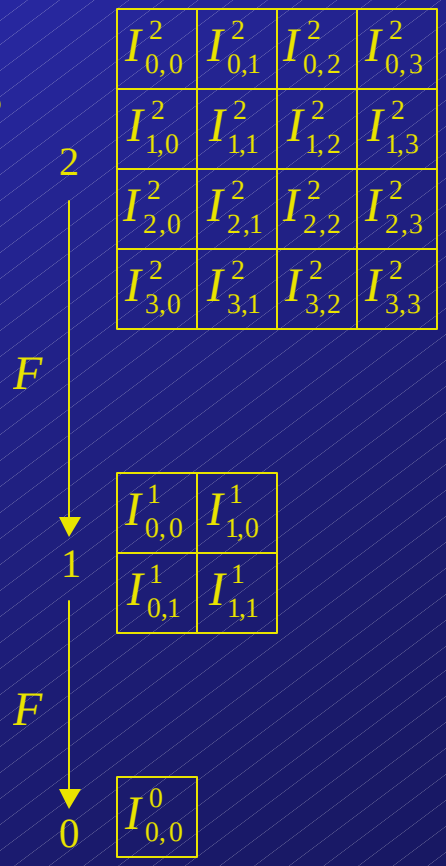
I - Progressive Texture Maps

- Computing the mipmap levels

Using low pass filter : $F = \begin{pmatrix} 0.25 & 0.25 \\ 0.25 & 0.25 \end{pmatrix}$

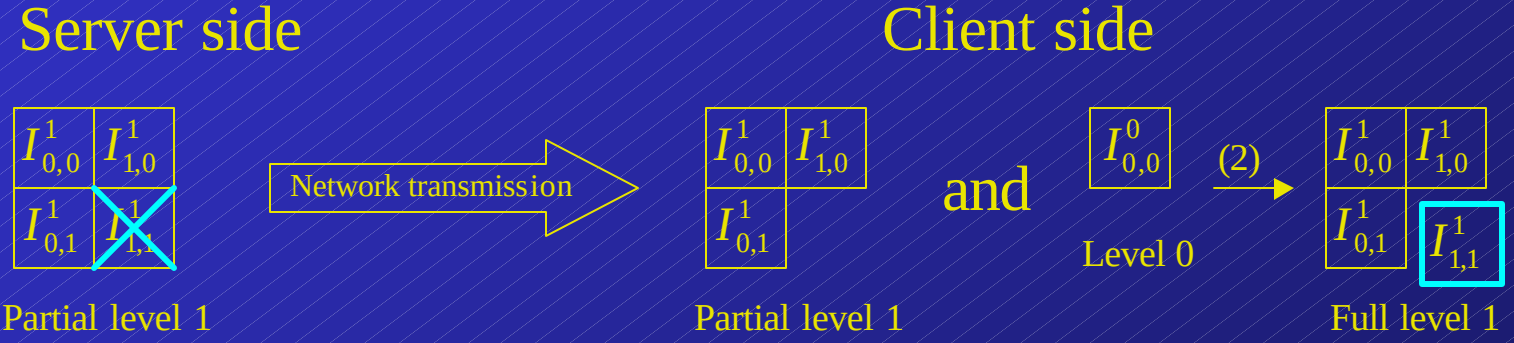
Color component for lower level :

(1) $$I_{i,j}^{n-1} = \frac{1}{4} \sum_{\substack{0 \le l \le 1 \\ 0 \le m \le 1}} I_{i.2+l,\, j.2+m}^{n}$$



I - Progressive Texture Maps

- Mipmap level reconstruction



$$(2)\quad I^n_{i.2+1,j.2+1} = 4.I^{n-1}_{i,j} - I^n_{i.2,j.2} - I^n_{i.2+1,j.2} - I^n_{i.2,j.2+1}$$

$\Rightarrow$ Mipmap levels memory size = original texture map size !!

I - Progressive Texture Maps

- Integer solution

$$(1)\quad I_{i,j}^{n-1} = \boxed{\frac{1}{4}} \sum_{\substack{0\le l\le 1 \\ 0\le m\le 1}} I_{i.2+l,\,j.2+m}^{n} \Rightarrow \text{remainder} \in \left\{0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}\right\}$$

$\Rightarrow$ Equation (2) becomes :

$$(3)\quad I_{i.2+1,\,j.2+1}^{n} = 4.I_{i,j}^{n-1} + \boxed{e_{i,j}^{n-1}} - I_{i.2,\,j.2}^{n} - I_{i.2+1,\,j.2}^{n} - I_{i.2,\,j.2+1}^{n}$$

With $e_{i,j}^{n-1} \in \{0,1,2,3\}$, stored using only 2 bits.

I - Progressive Texture Maps

- PTM
  - + Hierarchical representation
    - ⇒ Possibility to encode texture mipmaps
  - + Losseless compression
    - ⇒ Average compression equal 2:1
  - + No data redundancy between levels
  - + Original size increased only by 6%

II - Average Coverage Hints

ACH

A metric to select the mipmap levels to download

II - Average Coverage Hints

[Teler et al.]

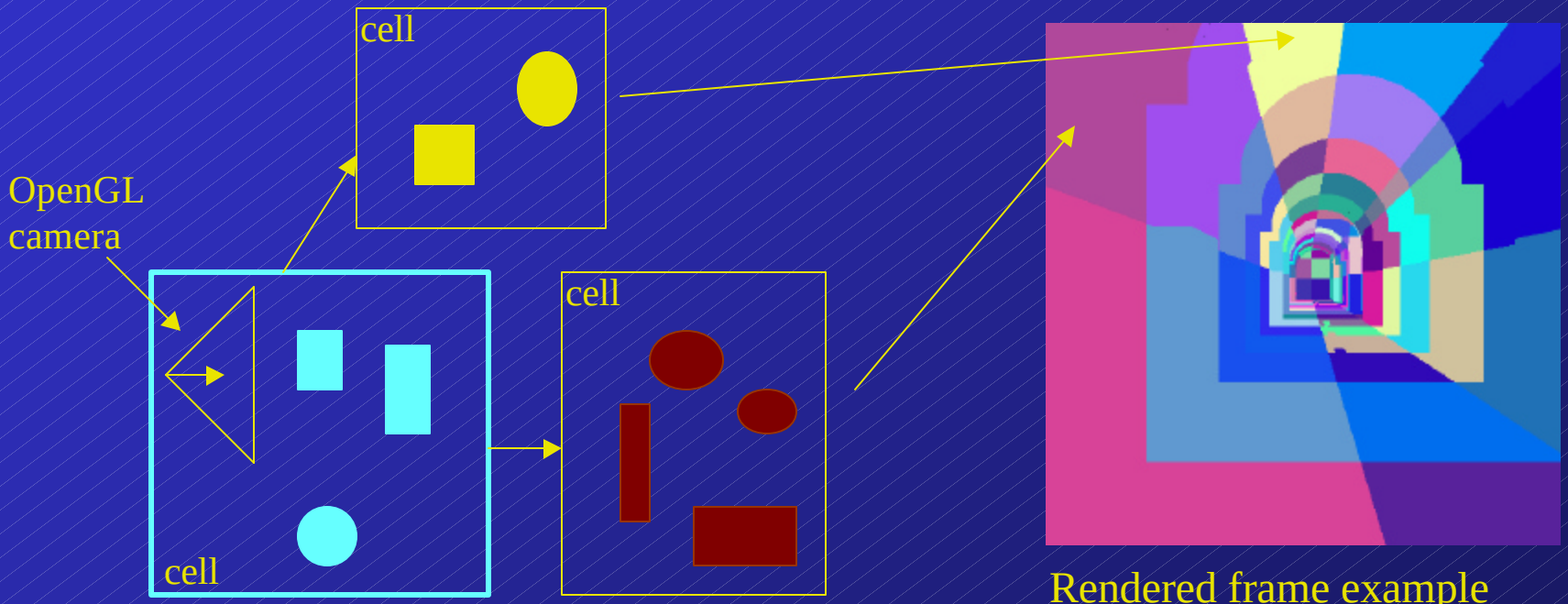
- Bounding box projection area
  - + Medium computation cost
  - – Occluded objects are treated as visible

[Dumont et al.]

- Frame buffer analysis (two pass)
  - + Precise, occluded objects are rejected
  - – Buffer analysis is very costly

II - Average Coverage Hints

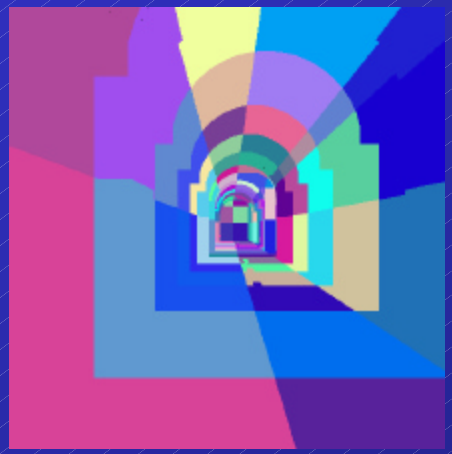
- Off-line : per-cell ACH computation



Rendered frame example

II - Average Coverage Hints

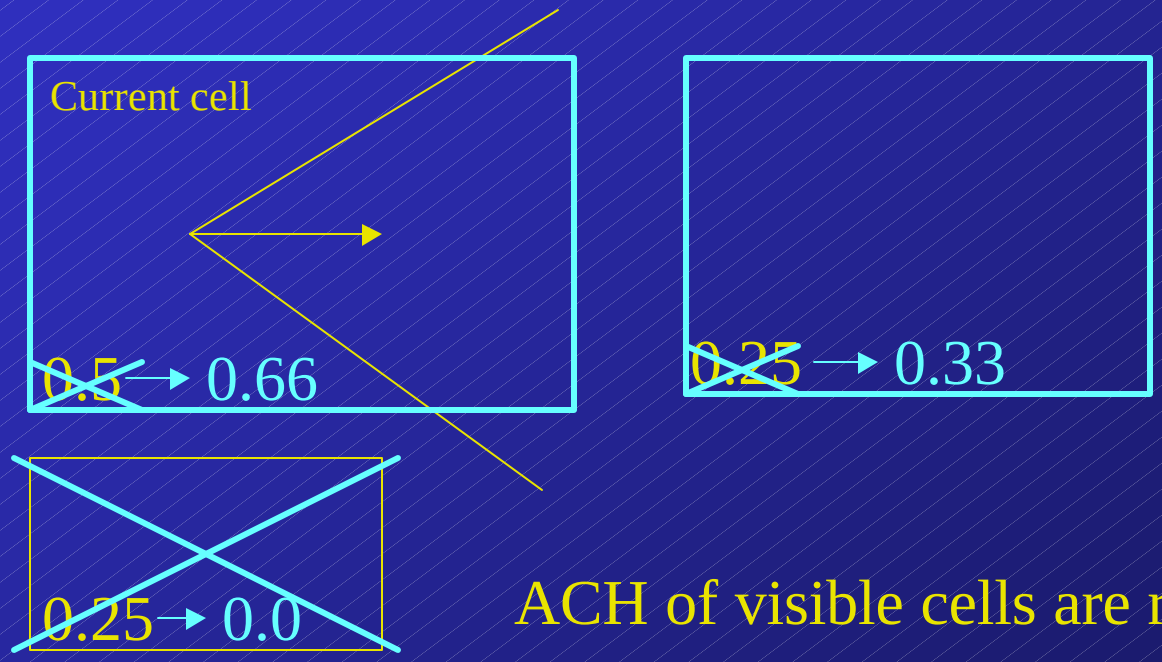
- Off-line : per-cell ACH computation
  - Height camera positions per cell,
  - Six directions for each position.



1. Pixel count for each color (cell),
2. Normalize values using total pixel number,
3. Get a percentage of coverage for each cell,
4. Store the result in the database.

II - Average Coverage Hints

- On-line : ACH renormalization



ACH of visible cells are renormalized

III - Automatic adaptation

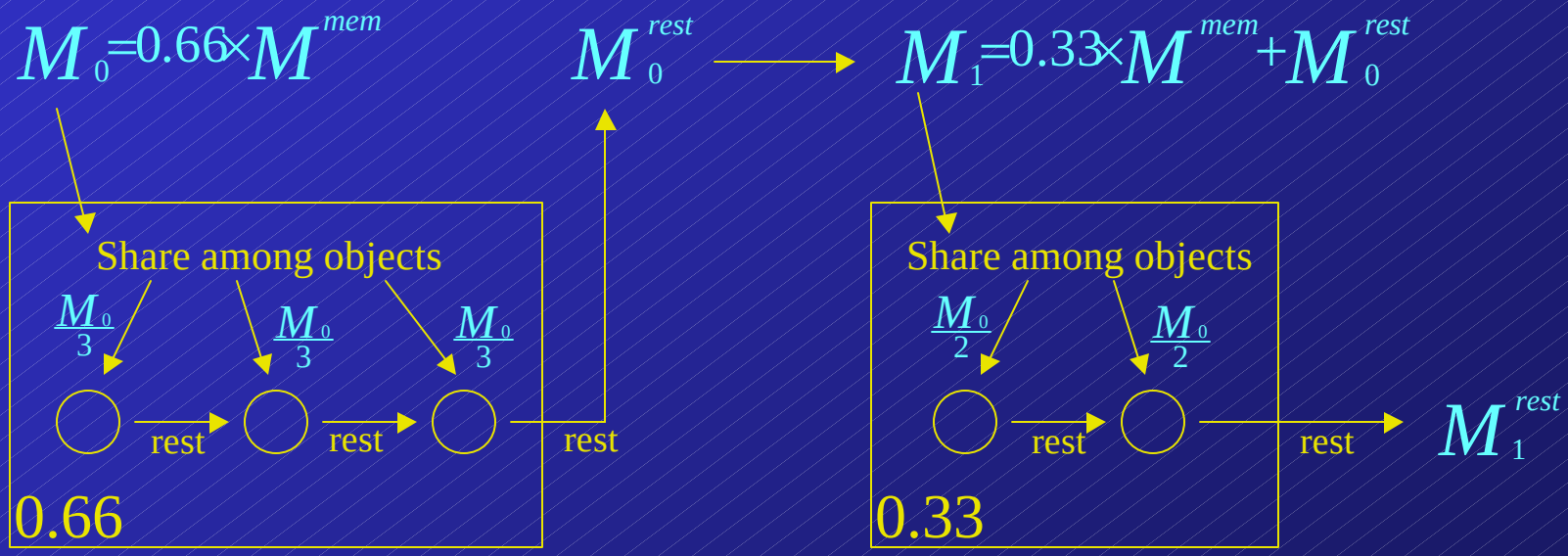
- Using PTMs and ACHs to provide
  - Highest visual quality
  - Maximum interactivity
- According to
  - Network bandwidth
  - Client machine power
  - User defined parameters

III - Automatic adaptation

- User defined parameters :
  - Amount of graphics hardware memory
    - Available for all the texture maps
      - When viewpoint is moving : M_{sta}^{mem}
      - When viewpoint is static more than n seconds : M_{dyn}^{mem}
  - Time limit to download a mipmap level
    - When viewpoint is moving : Δ_t^{dyn}
    - When viewpoint is static more than n seconds : Δ_t^{sta}

III - Automatic adaptation

- Memory budget allocation



Visible cells sorted according to their ACH value

III - Automatic adaptation

- Memory budget usage

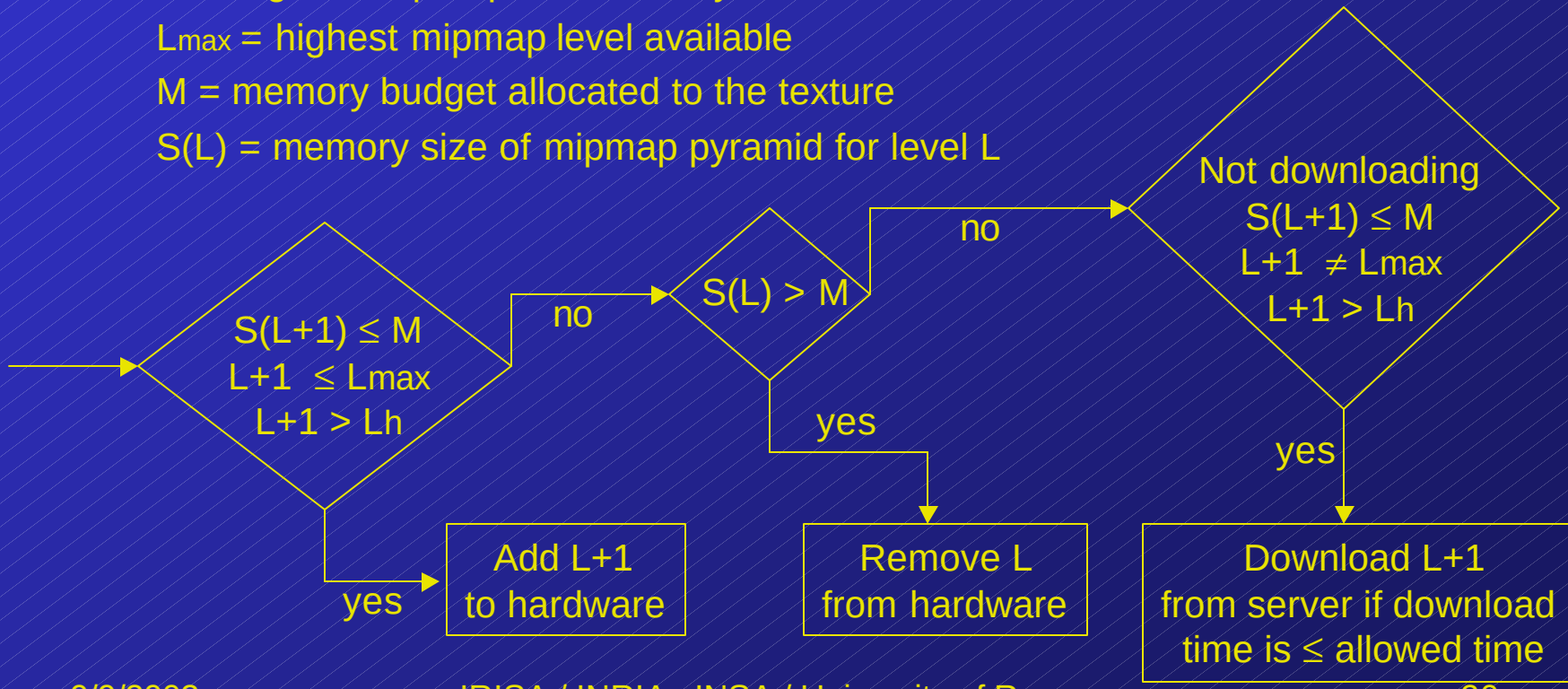
L = highest mipmap level into graphics hardware

L_h = highest mipmap level already downloaded

L_{max} = highest mipmap level available

M = memory budget allocated to the texture

S(L) = memory size of mipmap pyramid for level L



IV - Results

- Video
  - Model with 90MB of texture maps,
  - Pentium IV 1.7GHz,
  - Nvidia Quadro 2 Pro 64MB
  - User parameters :

$\Delta_t^{dyn} = 5sec$ $M_{dyn}^{mem} = 4MB$ $n = 5sec$

$\Delta_t^{dyn} = 20sec$ $M_{sta}^{mem} = 12MB$

Bandwidth simulation
at
128Kb/s, 256Kb/s
1Mb/s, 150Mb/s

Conclusion

- PTMs dedicated to texture mipmaps
  - Better loss less compression than JPEG
  - Low CPU cost for the decompression
- ACHs dedicated to conservative visibility
  - Preprocessed, low cost storage
  - Low CPU cost on the client side
- Automatic adaptation
  - Network and Client machine

Questions ?