Walking through images

Kadi Bouatouch
IRISA
Université de Rennes I, France

Goal

Rendering Complex Scenes on Mobile Terminals or on the web

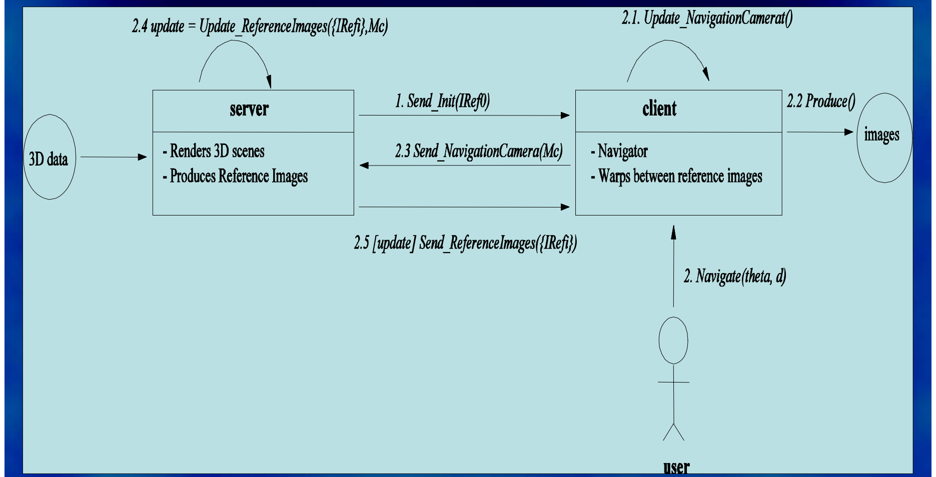
Rendering on Mobile Terminals

- IBR methods: rendering of complex scenes on a mobile terminal
- Mobile terminals: PDA, cell phone
- While using a client/server architecture.
- A PDA or a cell phone or a PC represents the client
- Server computes a very small set of key images
- Client utilizes these images to use a warping technique to compute new images as seen by intermediate cameras
- Intermediate cameras: positions and directions are chosen interactively by the user by moving the stylus of a PDA.

Rendering on Mobile Terminals

- Most difficult problem is how to place the cameras capturing the key images
- Camera placement allows an efficient warping avoiding artifacts, such as holes,
- Holes due to occlusions and exposures.
- Providing a general solution to the problem of camera placement is a hard task.
- We addressed only the case of urban scenes.

Rendering on Mobile Terminals



Rendering on Mobile Terminals

- Camera placement: a very difficult task
- No satisfying solution available
- One Solution
  - From a single image
    - reconstruction of a coarse 3D model
  - From a set of images of a real or virtual scene taken when walking through this scene
    - Coarse 3D model for each image
    - Put in correspondence the obtained 3D models: use geometry warping
  - Rendering

From a single image

ATIP

A Tool for 3D Navigation inside a Single Image with Automatic Camera Calibration

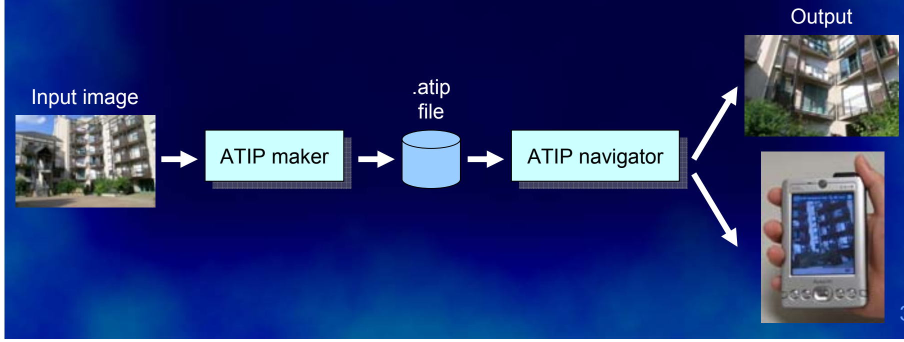
Kévin Boulanger, Kadi Bouatouch, Sumanta Pattanaik
IRISA, Université de Rennes I, France
University of Central Florida, USA

Purpose

Single image → Simple 3D navigation

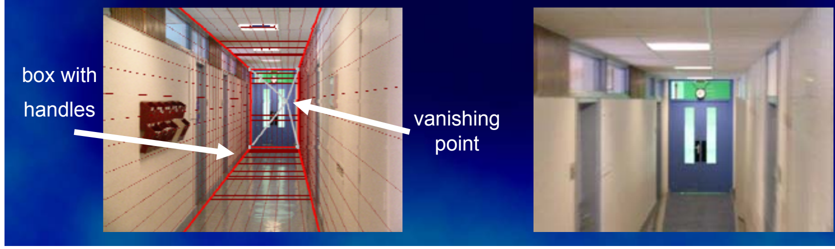
Software architecture

- *ATIP Maker* : processes the input image, creates an *.atip* file
- *ATIP Navigator* : reads the *.atip* file, allows multi-platform 3D navigation



Tour Into the Picture

- Original method by Horry et al.
  - Given an input image, the perspective effect is determined by manually (mouse click) choosing a point in the image as a vanishing point
  - The scene is manually approximated by a box by dragging its corner handles
  - Textures are computed for each box's face from the input image
  - The textured box is rendered from another viewpoint



Tour Into the Picture

- Manual fitting of the box
  - long and tedious
- Limitations
  - supports only images with a single vanishing point
  - horizontal and vertical lines must be parallel to the image borders
  - careful capture of the input image is necessary (say: using a tripod)

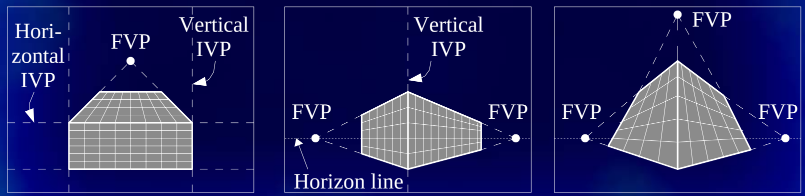
Automatic Tour Into the Picture

Contributions of ATIP:

- minimal user interaction
- more accurate calibration of the camera using multiple vanishing points
- input image from any orientation of the camera is acceptable
  - the camera can be hand-held

Vanishing points

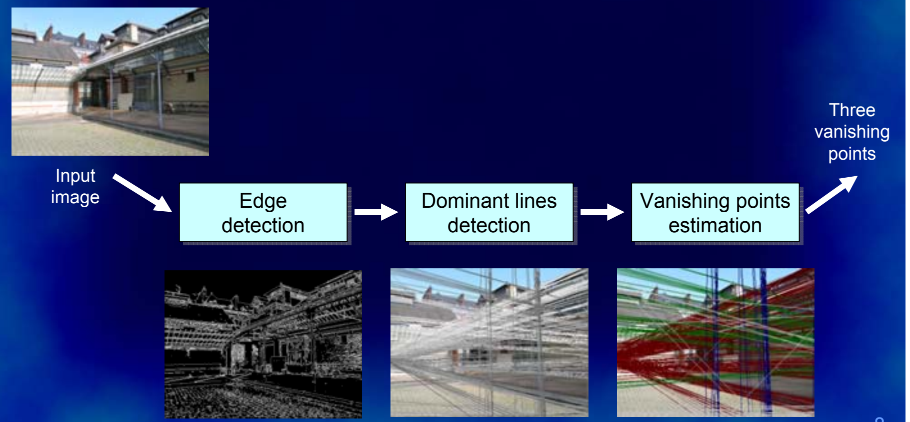
- Vanishing points help to calibrate the camera
- Always three vanishing points in an image
- Three combinations of finite and infinite vanishing points
- TIP manages only one finite vanishing point
- ATIP manages every situation, even with rotation around the view direction (very common without tripod)



1 finite vanishing point, 2 infinite vanishing points | 2 finite vanishing points, 1 infinite vanishing point | 3 finite vanishing points

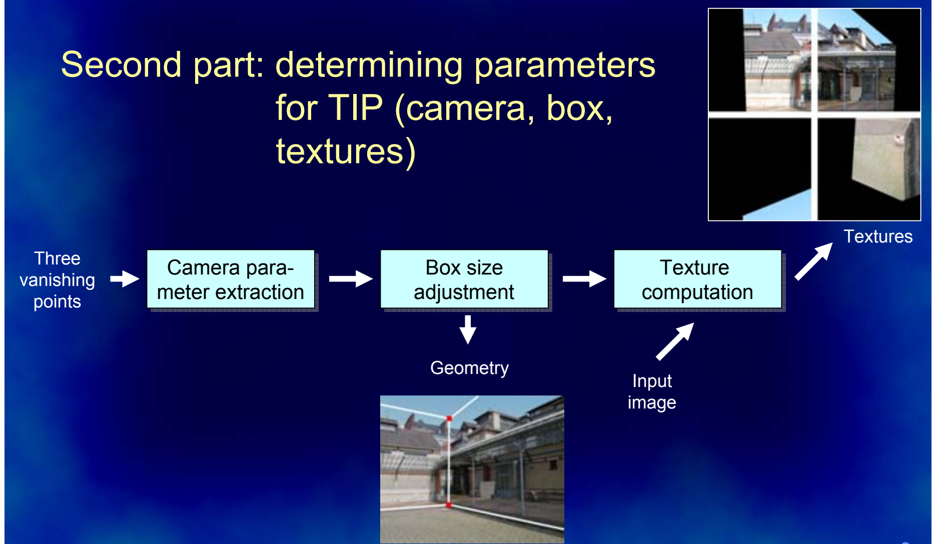
ATIP Maker pipeline

First part: image processing



ATIP Maker pipeline

Second part: determining parameters for TIP (camera, box, textures)



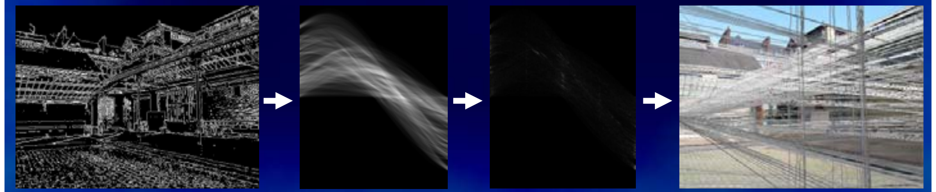
Edge detection

- Conversion to grayscale of the input image
- Logical AND between gradient-based and laplacian-based methods

Input image | Grayscale image | Edges

Dominant lines detection

- Dominant lines: longest alignments of points
- Vanishing lines = subset of the set of dominant lines
- How:
  - 1-to-m Hough transform of the edge points
  - high-pass filter of the Hough transform result
  - detection of the maxima by thresholding
  - transform the maxima back to lines in image space



Edges → Hough transform → High-pass filter → Dominant lines

Dominant lines detection
1-to-m Hough transform

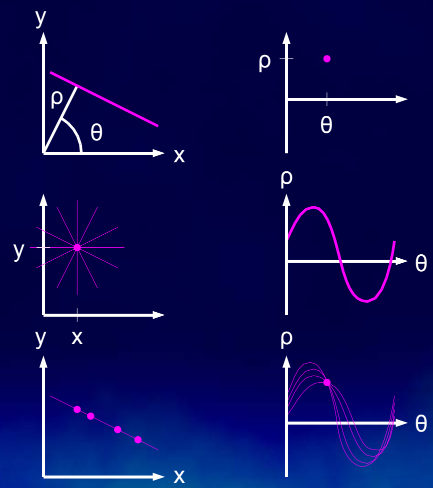


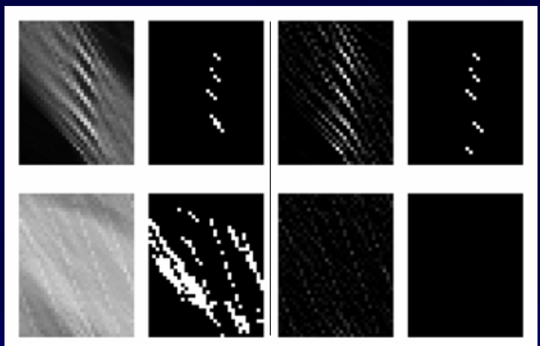
	Image space	Hough space	
Line (with polar parametrization)			Point
Point (intersection of an infinite number of lines)			Curve $\rho = x.\cos(\theta) + y.\sin(\theta)$
Aligned points			Set of curves with a common point, corresponding to a set of aligned points in image space

Dominant lines detection
Hough space high-pass filter

- Simple thresholding of the Hough transform
  → lot of noise due to strongly textured regions of the input image
- High-pass filter to remove the dense regions of high values (noise)

Region with points to be kept

Dense region of high values to be eliminated by filtering



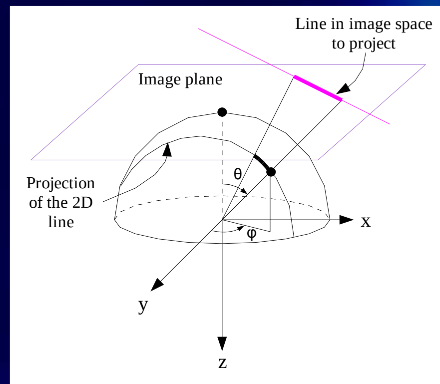
Kept points

Noise eliminated

Hough transform | Simple thresholding | High-pass filter | Final result

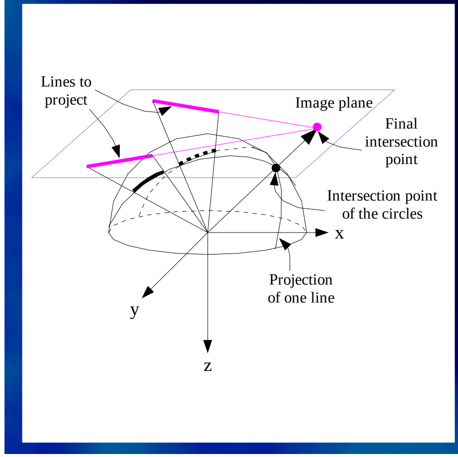
Vanishing points estimation

- Vanishing point = intersection of a subset of dominant lines
- Projection of each dominant line onto a hemisphere
- Accumulation of the resulting curves on the hemisphere
- Retrieve the three maxima
- Project them back to image space to get the vanishing point coordinates

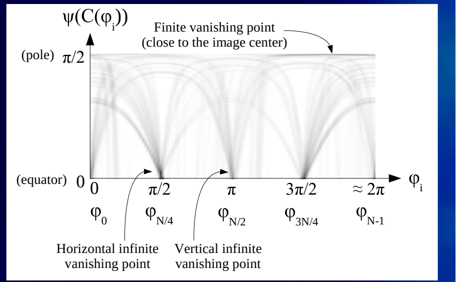


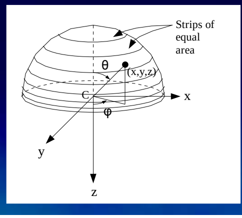
Vanishing points estimation
Accumulation

The intersection of curves on the hemisphere corresponds to the intersection of the corresponding lines in image space



Grid of accumulators





Non-uniform subdivision of the hemisphere

Vanishing points estimation
Three maxima retrieval

The three maxima cannot be retrieved in a single step, instead :

```
for i = 0 to 2
{
   filteredCells = low_pass_filter(hemisphere_cells);
   maxCell = maximum_value_cell(filteredCells);
   vanPoints[i] = project_to_image_plane(maxCell);
   dominantLines[i] = dominant_lines_associated_with(vanPoints[i]);
   hemisphere_cells = negative_accumulation(dominantLines[i],
                                            hemisphere_cells)
}
```

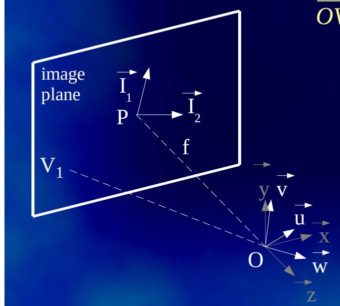
- Hypothesis: the three resulting vanishing points correspond to orthogonal directions
- With this algorithm, only dominant lines that contribute to the detection of vanishing points are considered

Camera parameter extraction

- From the three vanishing points, finite or infinite, extract camera parameters
  - Focal length and rotation matrix
- The three vanishing point directions define the world coordinate frame
- Process the three combinations of vanishing points different ways

Camera parameter extraction
One finite VP, two infinite VPs

- V_1 finite vanishing point
- $\vec{I}_1$ and $\vec{I}_2$ directions of the infinite vanishing points in image space
- O position of the camera
- $\vec{x}, \vec{y}, \vec{z}$ coordinate frame of the camera (fixed)
- f (focal length) is fixed by the user (48° by default)
- $\vec{u}, \vec{v}, \vec{w}$ world coordinate frame (columns of the rotation matrix), to be found



$$\overrightarrow{OV_1} = (V_{1x}, V_{1y}, -f)^T \qquad \vec{I}_1 = (I_{1x}, I_{1y}, 0)^T \qquad \vec{I}_2 = (I_{2x}, I_{2y}, 0)^T$$

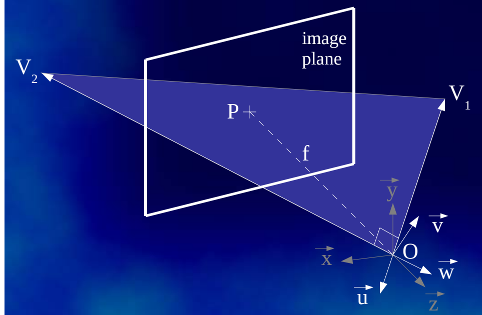
$$\vec{u}' = \left(I_{1x}, I_{1y}, \frac{I_{1x}V_{1x} + I_{1y}V_{1y}}{f}\right)^T \qquad \vec{u} = \frac{\vec{u}'}{\|\vec{u}'\|}$$

$$\vec{v}' = \left(I_{2x}, I_{2y}, \frac{I_{2x}V_{1x} + I_{2y}V_{1y}}{f}\right)^T \qquad \vec{v} = \frac{\vec{v}'}{\|\vec{v}'\|}$$

$$\vec{w}' = (-V_{1x}, -V_{1y}, f)^T \qquad \vec{w} = \frac{\vec{w}'}{\|\vec{w}'\|}$$

Camera parameter extraction
Two finite VPs, one infinite VP

- Two known VPs (V_1 and V_2), represent orthogonal directions
- f can be computed
- $\vec{v}$ obtained by cross product



$$\overrightarrow{OV_1} = (V_{1x}, V_{1y}, -f)^T$$

$$\overrightarrow{OV_2} = (V_{2x}, V_{2y}, -f)^T$$

$$\overrightarrow{OV_1}.\overrightarrow{OV_2} = 0 \qquad f > 0$$

$$\Rightarrow \; f = \sqrt{|V_{1x}V_{1y} + V_{2x}V_{2y}|}$$

$$\vec{u} = -\frac{\overrightarrow{OV_1}}{\|\overrightarrow{OV_1}\|} \qquad \vec{w} = -\frac{\overrightarrow{OV_2}}{\|\overrightarrow{OV_2}\|}$$

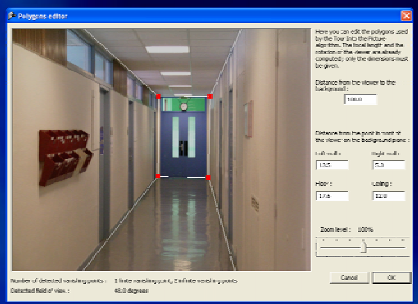
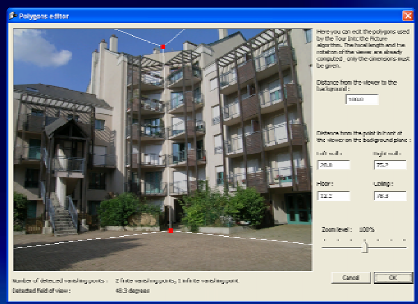
$$\vec{v} = \vec{w} \times \vec{u}$$

Camera parameter extraction
Three finite vanishing points

- Three finite vanishing points, $\vec{v}_1$, $\vec{v}_2$ and $\vec{v}_3$
  → overconstrained problem
- The two finite VPs method used for $(\vec{v}_1, \vec{v}_2)$, $(\vec{v}_2, \vec{v}_3)$ and $(\vec{v}_3, \vec{v}_1)$
- The dot product between the third VP direction and the cross product of the two first directions gives an error value
- Choose the set of vectors giving minimal error

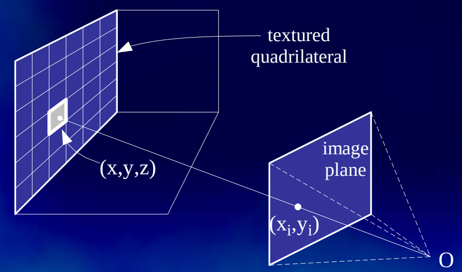
Box size adjustment

- The only step of the algorithm that requires user interaction
- Simpler than TIP since the camera is calibrated, only the corners (red handles) have to be moved

Texture computation

- Data retrieved from the input image
- Each box's face is subdivided into a uniform grid, each cell corresponds to a texel
- Each texel center point of a texture is projected onto the input image plane
- Bilinear interpolation gives the final color
- Points falling outside the input image bounds are set to black





Results

Results

Results with difficult scenes

Forest scene:
Robust dominant lines detection from noisy edges using high-pass filter on Hough transform

Non-flat surfaces:
Robust «maxima on the hemisphere» computation algorithm

Hand-drawn sketch:
Correct detection of dominant lines from noisy edges and imprecise intersections of vanishing lines

From multiple images

- Solution
  - Choose one camera frame CF
  - Determine the transformations between CF and another camera frame ACF
  - Transform the 3D model, found for one ACF, to CF
  - Merge the two models, result = combined model CM
  - CM expressed in CF frame + CM expressed in ACF frame
  - For an intermediate camera, interpolate between each pair of corresponding vertices: geometry and textures

ATIP with multiple images

- Solution
  - Server: sends the client the combined 3D models associated with two or more successive images captured by the camera for different positions and orientations
  - One 3D model = N faces + N textures
  - Client: renders the received 3D models as well as in-between interpolated 3D models

Questions

- http://www.irisa.fr/siames/Kevin.Boulanger
- Kévin Boulanger, Kadi Bouatouch, Sumanta Pattanaik

  ATIP: A Tool for 3D Navigation inside a Single Image with Automatic Camera Calibration. EGUK 2006, june 2006
- Gwenola Thomas, Gérald Point, Kadi Bouatouch

  A Client-Server Approach to Image-Based Rendering on Mobile Terminals

  Technical Report RR-5447, INRIA - 2005

Rendering on Mobile Terminals A Client-Server Approach to IBR

- The IBR process is initialized when the server sends the client an initial reference image together with its corresponding camera parameters (1.Send_Init(IRef0)).
- IRef = a reference image and its corresponding camera parameters.
- On the client side, the user can navigate through the 3D environment by changing the orientation and the position of the camera (2.Navigate(theta,d)).
- In the present application, navigation is performed in a urban environment.
- The position of the camera is constrained to lie on a horizontal plane. two successive images to make the IBR approach possible.
- These limitations are also coherent with the way people walk in a city.

Rendering on Mobile Terminals A Client-Server Approach to IBR

- Whenever the user moves the navigation camera, the client computes a new image by warping some of the available reference images (2.1.Update_NavigationCamera(), 2.2.Produce()).
- Available reference images: not always appropriate for warping,
- Available reference images too far from the current navigation camera may cause the appearance of holes on the warped image.
- We use blurring filtters to fill the appeared holes.
- To maintain an appropriate set of reference images on the client side, the client transmits the parameters of the new current navigation camera to the server whenever the user moves the camera (2.3.Send_NavigationCamera(Mc)).

Rendering on Mobile Terminals A Client-Server Approach to IBR

- The set of reference images, available on the client side, is appropriate if each reference image of this set significantly contributes to the construction of the warped image.
- The contribution of a reference image is measured as the percentage of pixels of the reference image that re-project onto the warped image

Rendering on Mobile Terminals A Client-Server Approach to IBR

- The server owns the 3D urban scene and a set of edges that define the geometry of the street network.
- Depending on the current navigation camera Mc on the client side and on the reference images previously sent IRefi, the server is able to determine whether the reference images, available on the client side, have to be updated or not (2.4.update=Update_ReferenceImages({IRefi})).
- A reference image has to be replaced on the client side when it does not signicantly contribute to the warped image.
- If necessary updates, the server sends the client new ref images (2.5.[update]Send_ReferenceImages({IRefi})).
- The way the cameras are positioned in the environment and the way the server selects them to compute reference images are provided by the camera placement algorithm.

Rendering on Mobile Terminals A Client-Server Approach to IBR

- The IBR process is initialized when the server sends the client an initial reference image together with its corresponding camera parameters (1.Send_Init(IRef0)).
- IRef = a reference image and its corresponding camera parameters.
- On the client side, the user can navigate through the 3D environment by changing the orientation and the position of the camera (2.Navigate(theta,d)).
- In the present application, navigation is performed in a urban environment.
- The position of the camera is constrained to lie on a horizontal plane. two successive images to make the IBR approach possible.
- These limitations are also coherent with the way people walk in a city.

Rendering on Mobile Terminals A Client-Server Approach to IBR

- Whenever the user moves the navigation camera, the client computes a new image by warping some of the available reference images (2.1.Update_NavigationCamera(), 2.2.Produce()).
- Available reference images: not always appropriate for warping,
- Available reference images too far from the current navigation camera may cause the appearance of holes on the warped image.
- We use blurring filtters to fill the appeared holes.
- To maintain an appropriate set of reference images on the client side, the client transmits the parameters of the new current navigation camera to the server whenever the user moves the camera (2.3.Send_NavigationCamera(Mc)).

Rendering on Mobile Terminals A Client-Server Approach to IBR

- The set of reference images, available on the client side, is appropriate if each reference image of this set significantly contributes to the construction of the warped image.
- The contribution of a reference image is measured as the percentage of pixels of the reference image that re-project onto the warped image

Rendering on Mobile Terminals A Client-Server Approach to IBR

- The server owns the 3D urban scene and a set of edges that define the geometry of the street network.
- Depending on the current navigation camera Mc on the client side and on the reference images previously sent IRefi, the server is able to determine whether the reference images, available on the client side, have to be updated or not (2.4.update=Update_ReferenceImages({IRefi})).
- A reference image has to be replaced on the client side when it does not signicantly contribute to the warped image.
- If necessary updates, the server sends the client new ref images (2.5.[update]Send_ReferenceImages({IRefi})).
- The way the cameras are positioned in the environment and the way the server selects them to compute reference images are provided by the camera placement algorithm.