Haptic Rendering: Depth Image-based Approach

Prof. Jeha Ryu

Human-Machine-Computer Interface Laboratory
Department of Mechatronics
Gwangju Institute of Science and Technology (GIST), KOREA

Motivation

- Accurate and Efficient Haptic Rendering Algorithm for 3-and 6-DOF haptic interaction
- For highly complex and large-scale VEs that may simultaneously contain different types of object data representations.
- does not require any hierarchical data structure or any data conversion for multiple data formats.
  - local object geometry information by six virtual cameras
  - six depth maps at each pixel buffer in rendering process
  - computations to find the IHIP and collision responses are performed in the CPU.

Previous Works

- Geometry-dependent haptic rendering algorithms
  - Dependent on the geometry representation of the object

1) Surface-based for triangular meshes, polygons, NURBS, implicit surfaces
2) Volume-based for volume data using voxels
3) Point-based for point clouds ???

- Hierarchical Database or Preprocessing for collision detection speed-up

1) Surface-based by bounding volume hierarchy (AABB, OBB)
2) Volume-based by spatial partitioning (Octree, BSP (Binary Space Partition) tree) with voxelization preprocessing
3) Point-based for point clouds ???

* Constructing sophisticated hierarchical data structure for every object in a very complex VE ➔ may be time consuming
* For objects with diverse data representations ➔ preparation of consistent datasets or different haptic rendering algorithms for each.

Previous Works

- **Volumetric haptic rendering algorithms --- McNeely (1999, 2006)**
  - simple, fast, approximated voxel-based for rigid-rigid contact (multiple DOF contact)
  - Static arbitrary large and complex VE is voxelized and stored in preprocessig stage
  - Point shell representing a 3D tool with feature points and normal vectors inward the tool for collision response force computation
  - Later improvements: spatial accuracy by augmenting standard voxel-sampling with distance field, temporal coherence, culling of redundant polyhedral surface interactions
  - Rigid-Deformable object contact by Barbic and James (2007)

- **Differences for the proposed depth-image-based algorithm:**

(i) On-the-fly voxelization only for local, small portion of the object in very complex and hybrid data sets

(ii) No prior data hierarchy

(iii) Hybrid CPU/GPU

(iv) Constraint-based

➔ More scalable to object data representations, cheap, efficient, and accurate !!!

Previous Works

- GPU-based OpenHaptics

1) Use buffers in GPU during haptic rendering
2) Use feedback buffer storing geometry primitives (vertex, edges, polygons)
 or Capture the local geometry on-the-fly by one camera using depth buffer in the graphics pipeline
4) Read the geometry from the buffer and compute response force with a proxy
➔ Not efficient for highly complex object and hybrid data representations

- Direct Use of GPU

1) 6-DOF for polygonal models (2003-2008, UNC)
➔ Need surface normal vectors and limited to convex primitives for both 3D tool and a contact object

Previous Works

- Image-based rendering (IBR) & collision detection
  - Use images as an alternative to traditional geometry-based techniques
  - IBR designed to handle dynamic scenes --- Video-based rendering
  - Independent of the object topology
  - Independent of the object complexity

- Depth image-based haptic rendering

1) LOMI --- local depth image-based voxelizaion, depth image-based contact query

2) DIBHR --- haptic interaction with 2.5D/3D video media captured by one (real) depth camera

Previous Works
– Image-based collision detection

- **RECODE --- Baciu et al., (1998)**
  - Transforms the problem of 3D collision detection to 1D interval test along the Z-axis
  - Easily test by using a mask in the stencil buffer based on one of the two objects coordinates

Sixth Pacific Conference on Computer Graphics and Applications, 1998, pp.125-133

- **Fast and Simple 2D collision test --- Hoff et al., (2001)**
  - Possible collision is detected by bounding boxes
  - Enables proximity queries to be made using the graphics hardware to detect overlapping polygons

I3D '01 Proceedings of the symposium on Interactive 3D graphics, 2001, pp.145-148

Previous Works
– Image-based collision detection

- **CULLIDE --- Govindaraju et al., (2003)**
  - Uses visibility queries to detect potentially colliding objects set (PCS)
  - Pruning to remove fully visible objects and to identify potential regions of each object in PCS that may be involved in collision detection

Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware, pp.25 – 32

- **LDI --- Heidelberger et al., (2003), Faure et al., (2008)**
  - Potentially colliding objects are detected by using AABB bounding volume
  - Multiple layers of geometry seen from one viewpoint (LDI) is created to represent the intersection volume between the two objects
  - Penalty forces is computed as repulsion forces that try to minimize the intersection volume

Proceedings of the Vision, Modeling, and Visualization Conference 2003, pp. 461–468
SCA '08 Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp.155–162

Previous Works
– Depth image-based haptic rendering

- **Depth Image-Based Haptic Rendering**

- Treat virtual object in image space by using the final rendered scene in depth buffer
- Replace a complex collision detection process with a simple process of successive static images
- As mentioned, it is independent of the object topology and the object complexity

- **Depth image-based haptic rendering**

1) LOMI --- Kim et al., (2004, 2009)

- local depth image-based voxelizaion, depth image-based contact query

Presence: Teleoperators and Virtual environments, vol.18, no.5, 2009, pp.340-360

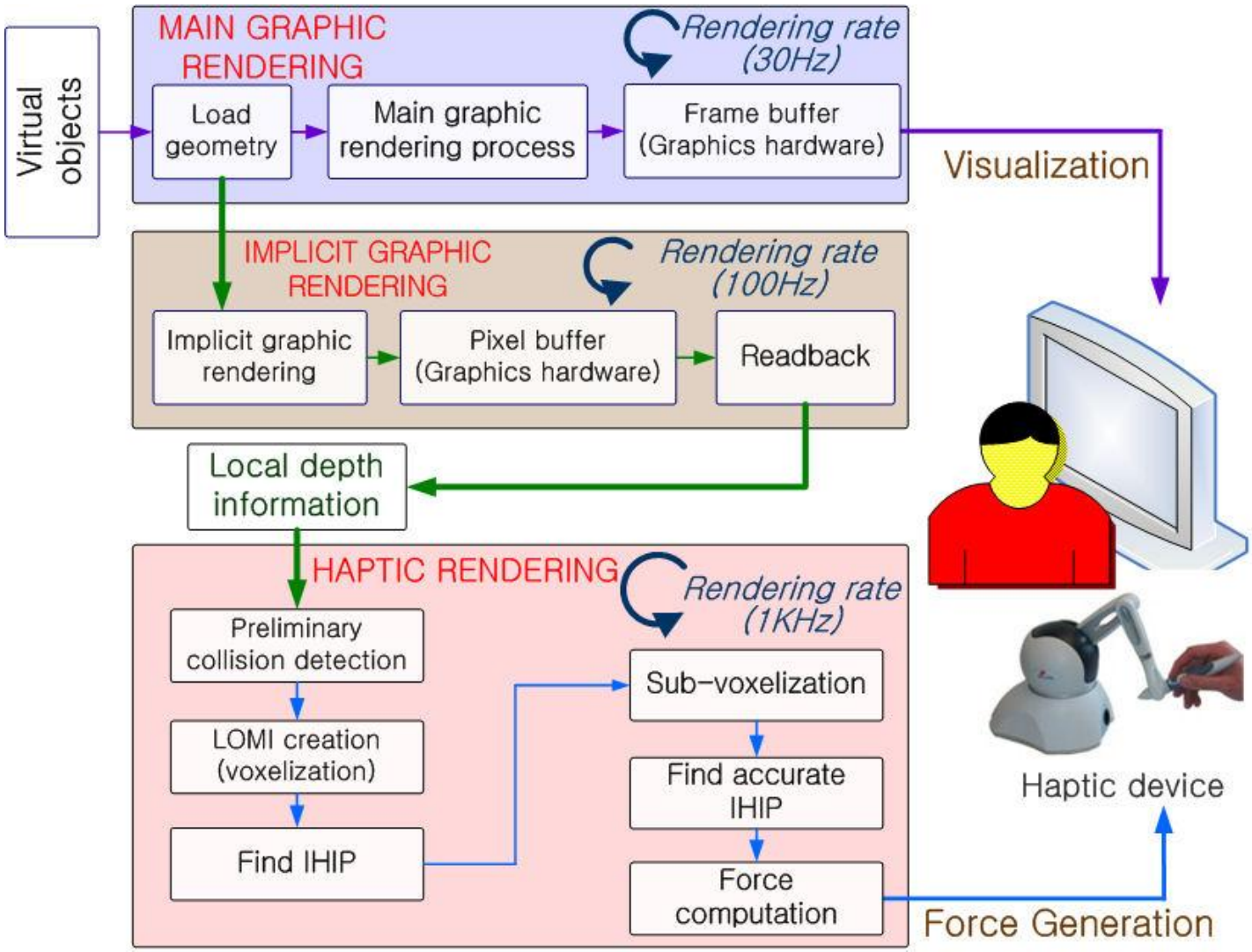
2) DIBHR --- Cha et al., (2005, 2006, 2008, 2009)

- Haptic interaction with 2.5D/3D video media captured by one (real) depth camera
- Haptic broadcasting system, Touchable 3D video system

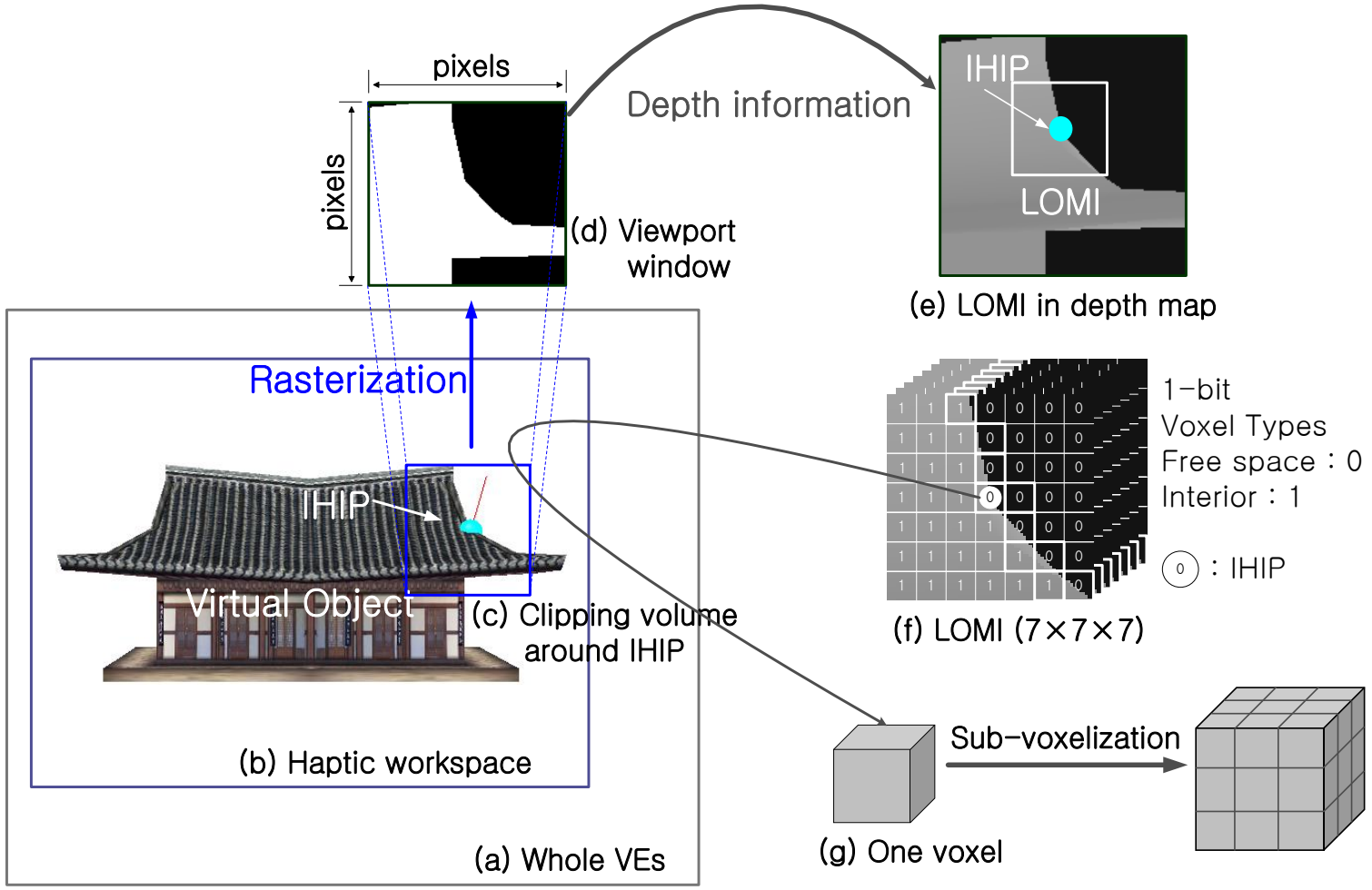
IEEE Transactions on Consumer Electronics, vol.52, issue.2, 2006, pp.477-484
Haptics: Perception, Devices and Scenarios, LNCS vol.5024, 2008, pp.640-650
IEEE Multimedia, vol.16, issue.3, 2009, pp.16-27
ACM Transactions on Multimedia Computing, Communications, and Applications, vol.5, issue.4, 2009, pp.1-25

Depth Image Based 3-DOF Haptic Rendering

Algorithm Overview



Algorithm Overview: Schematic Representation



Algorithm Overview

- In order to position the virtual cameras, define a haptic workspace as a cube enclosing the object to be touched
- A clipping volume containing a portion of the object around the IHIP is then selected
- The clipping volume of the object is implicitly rasterized in the GPU
  - non-visible off-screen rendering in each viewport window
  - six directional depth maps are acquired from the graphics pipeline using a read back process.
  - depth maps are used for collision detection
  - a LOMI is abstractly formed in the depth maps

Algorithm Overview

- The abstract LOMI is centered at the preliminary collision position
  - LOMI is represented using 1-bit voxels as a rectangular 3D grid structure comprised of two voxel types
  - abstract LOMI does not allocate any physical memory to store the voxel types

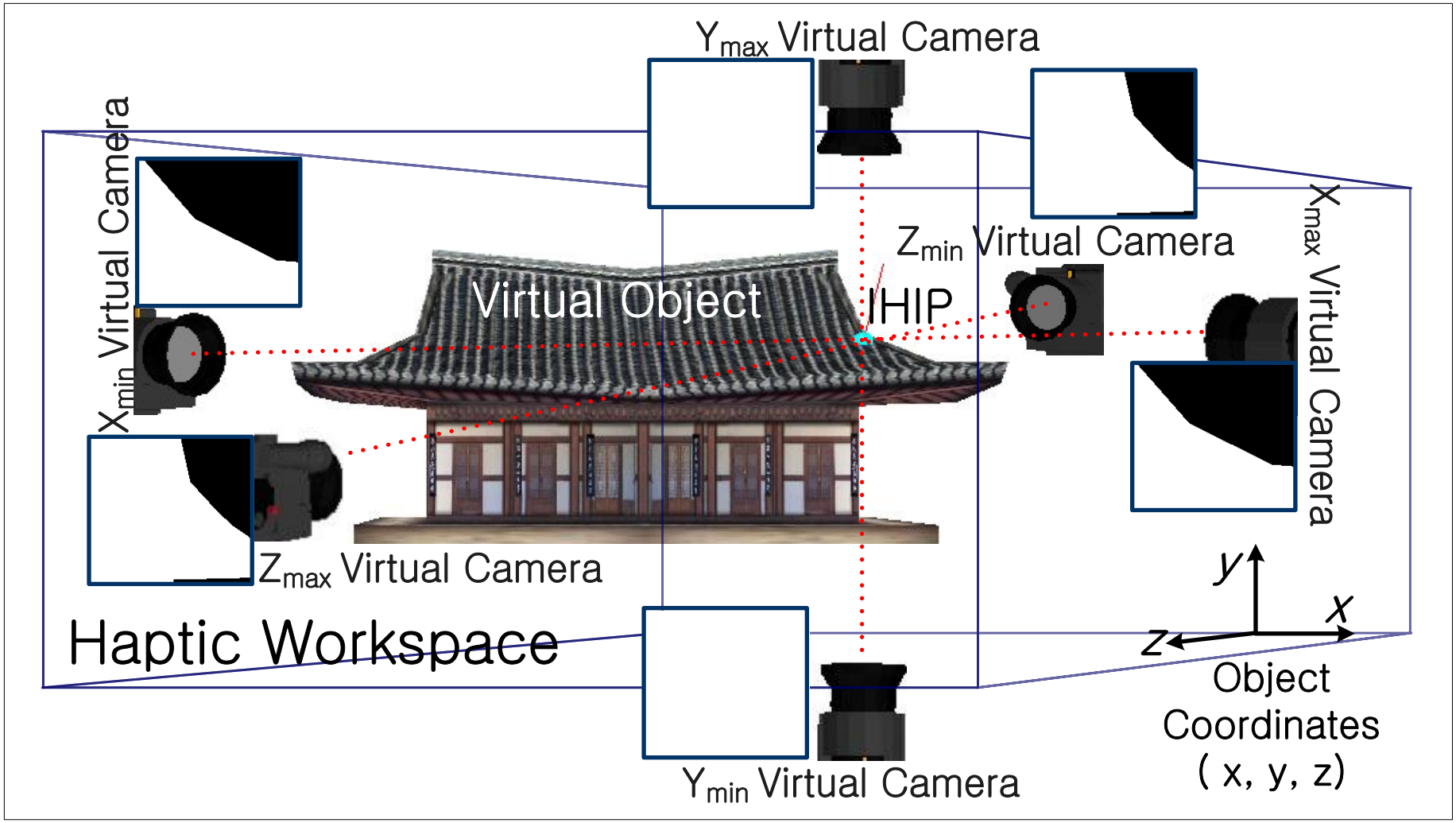
- The IHIP voxel can then be further divided into three regions on each side in order to achieve a higher resolution of the IHIP location.
  - One exterior voxel among the sub-voxelized voxels can then be selected as a more accurate IHIP.
  - This sub-voxelization process is then repeated until the actual desired minimum resolution.

- The collision response force can easily be computed the directional vector from the HIP to the IHIP.

Systematic Design Guidelines

- Haptic workspace for tangible object selection
- Clipping volume for tangible region selection
- Viewport resolution for desired haptic resolution
- Projection for uniform depth resolution
- LOMI size for true IHIP location

Selection of Haptic Workspace for Tangible Object
(for additional graphic rendering)



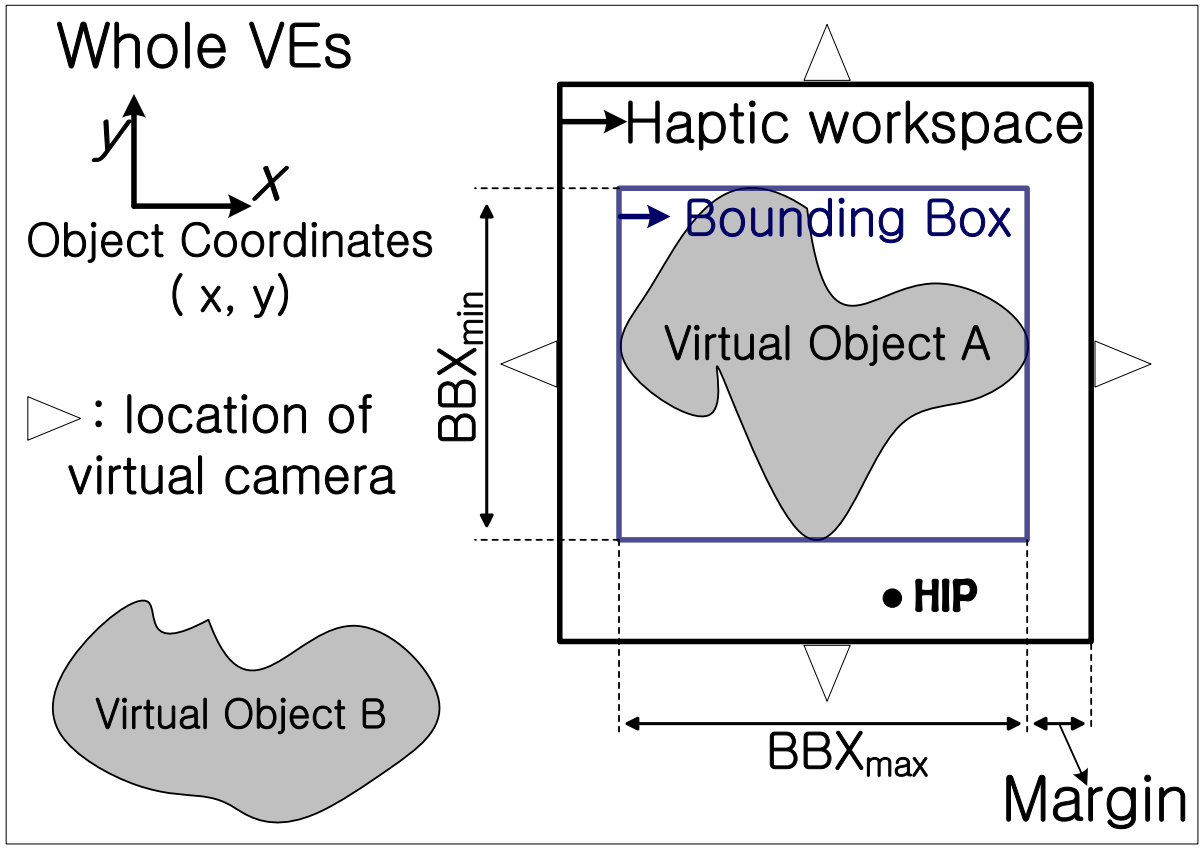
Selection of Haptic Workspace for Tangible Object (for additional graphic rendering)

- Figure shows the six virtual cameras and their respective small windows on the surfaces of a cubic haptic workspace.

- HIP movement is physically limited inside the haptic workspace when a user interacts with a virtual object
  - the virtual cameras should be located on each orthographic surface of a haptic workspace

- The small windows represent orthographic rendering by each virtual camera
  - the white color denotes the rendered portion of an object.

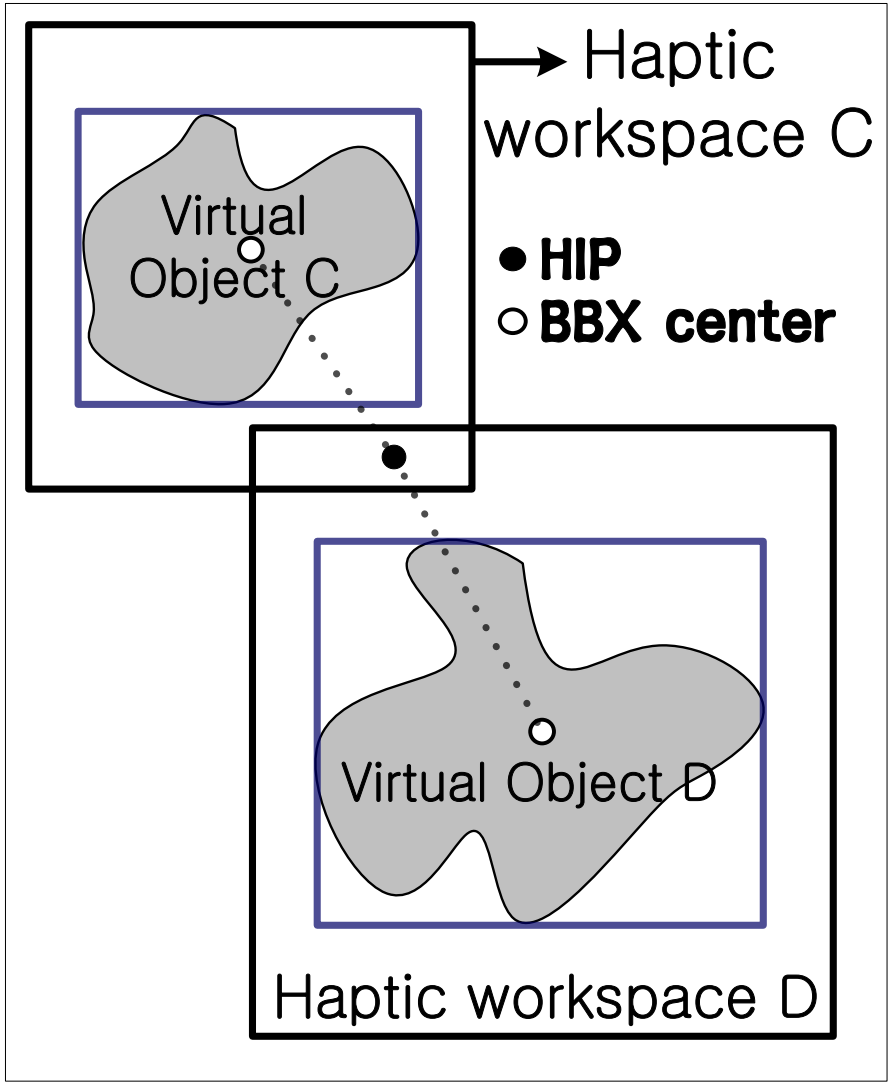
Selection of Haptic Workspace for Tangible Object : Size



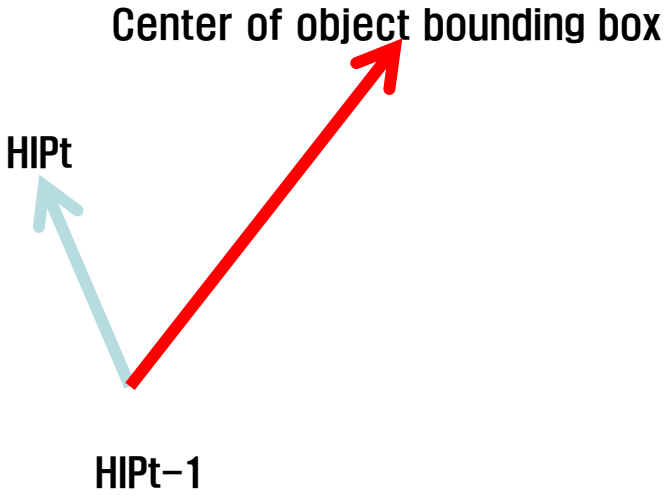
$$CubeSize = MaxDimension + Margin$$

$$Margin \geq \frac{2 \times DeviceSpeed_{\max}}{GraphicRate_{additional}}$$

Selection of Haptic Workspace for Tangible Object : Occlusion Avoidance



For convex objects, consider object trajectory together with current HIP position:



For concave objects, decompose into several convex objects.

Selection of Haptic Workspace for Tangible Object

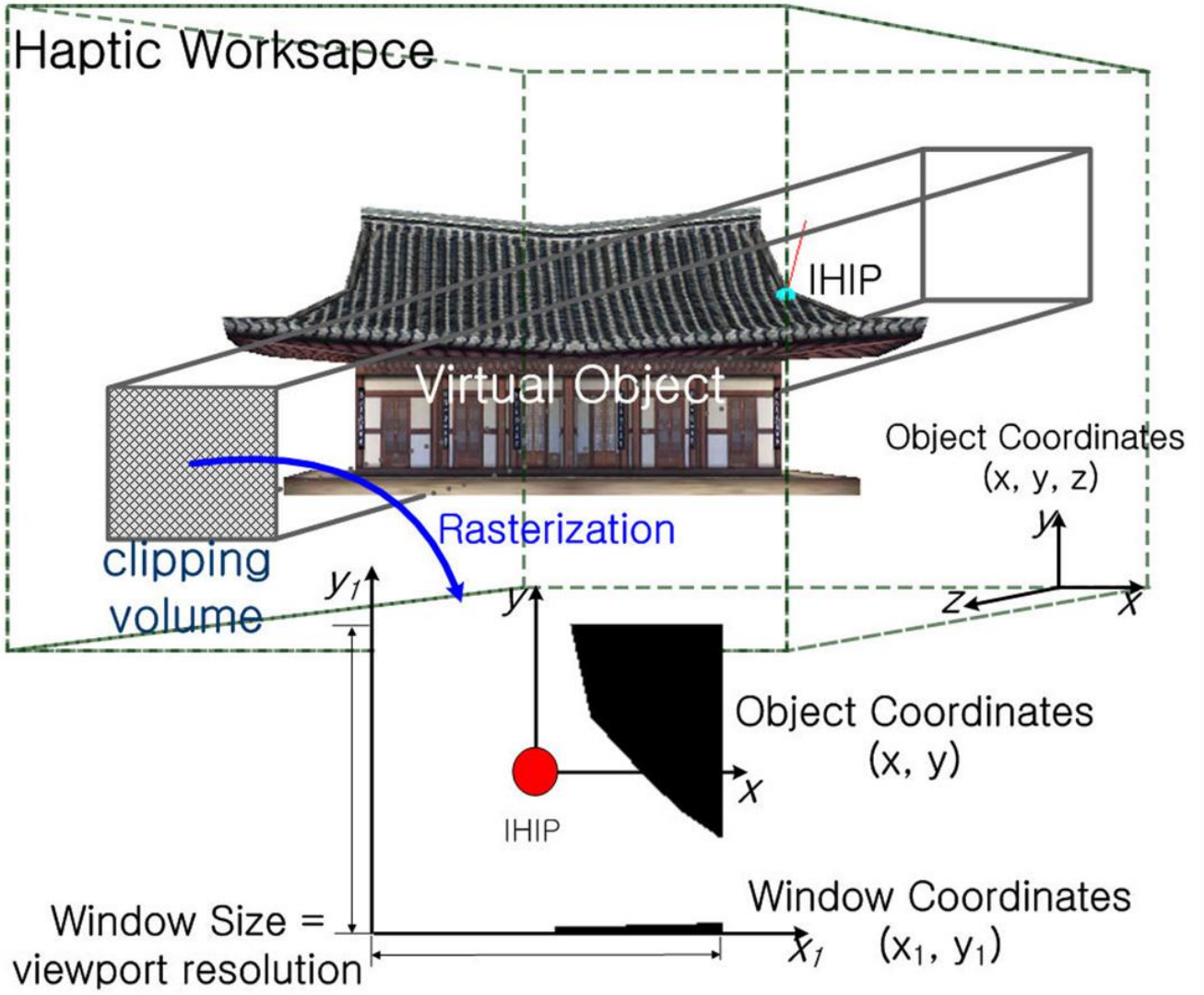
- To avoid occlusion problems when multiple objects are lined up in an orthographic viewing direction
  - the haptic workspace should be adaptively selected as a bounding cube

$$CubeSize = MaxDimension + Margin$$

$$Margin \geq \frac{2 \times DeviceSpeed_{\max}}{GraphicRate_{additional}}$$

- *MaxDimension* : the maximum bounding box size enclosing an object
- *Margin* : no depth information can be obtained if the virtual camera is attached directly to the surface of the object
- 2 × : the device can move in positive and negative coordinate directions from the origin
- $DeviceSpeed_{max}$: user must select a nonzero that is not the actual on-line device speed while interacting with a object.
- $GraphicRate_{additional}$: It is the rendering time of the six implicit renderings,

Selection of Clipping Volume for Tangible Region



Selection of Clipping Volume for Tangible Region

- Clipping volume is defined in the object coordinates, and explains how large portions of objects are rendered in the rendering window by the six virtual cameras.

- the center of the clipping volume is the position of the IHIP
  - the center of the clipping volume must always follow the IHIP position.

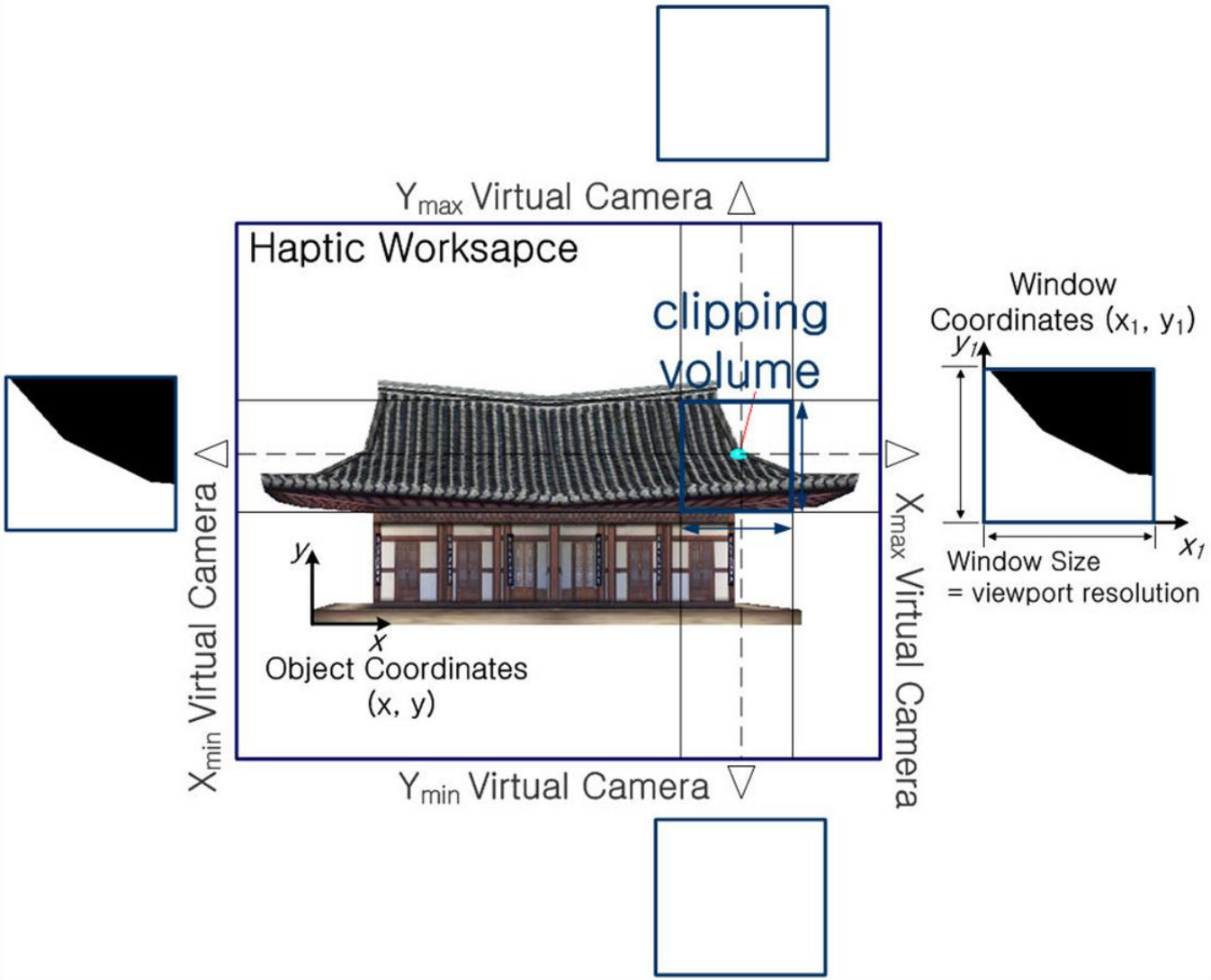
- Clipping volume size should be large enough to capture any IHIP movement between the successive additional graphic renderings
  - the clipping volume should always include an IHIP at the center

$$ClipVolumeSize \geq \frac{2 \times DeviceSpeed_{\max}}{GraphicRate_{additional}}$$

- Therefore, Margin = ClipVolumeSize

  - $GraphicRate_{additional}$: It is the rendering time of the six implicit renderings,

Viewport Resolution for Desired Haptic Resolution



Viewport Resolution for Desired Haptic Resolution

- the viewport resolution indicates how many pixels are assigned in the rendering window.

- The clipping volume is rasterized into each window of the virtual cameras in the graphics pipeline.
  - the clipping volume is containing a portion for the object.

$$ViewportResolution \geq \frac{ClipVolumeSize}{DesiredHapticResolution}$$

  - *DesiredHapticResolution* : the minimal meaningful surface geometric feature that can be haptically displayed to users
  - one pixel in a rendering window represents the physical size of the local geometry in the object coordinate
  - Viewport Resolution = Clipping Volume Window Size in # of pixels

Orthographic Projection for Uniform Depth Resolution

- In order to ensure a uniform depth resolution along the distance to the object, orthographic projection can be used
- With perspective projection:
  - depth information is nonlinear with respect to the distance between a virtual camera and the object rendered in the VE, i.e., depth resolution is coarse when the object is located far from the virtual camera.
  - the depth resolution is dependent both on the size of the haptic workspace and on the depth buffer precision of GPU.

$$DepthResolution = \frac{Distance}{DepthBufferPrecision}$$

- Distance between two cameras along the viewing direction

- Therefore, if a haptic workspace is 50 cm, the depth resolution becomes 0.05 μm for 24-bit depth buffer precision. This means that most collisions can be detected by using a 1-μm resolution.

LOMI Size for True IHIP Location

- Additional graphic rendering on the GPU can successfully provide local geometry information required for accurate and efficient haptic feedback on the CPU.
- The LOMI size can be decided based on both the speed of the probe and on the haptic rendering rate

$$LOMIsize \geq \frac{2 \times DeviceSpeed_{\max}}{HapticRenderingRate}$$

  - To correctly establish the LOMI and thereby find the true IHIP, this maximum speed must be determined such that any real speed is within the maximum allowable speed.
  - 2 mm for 1 kHz of haptic rendering & 1 m/sec speed of haptic interaction
  - voxel size = desired haptic resolution
- The number of voxels can then be calculated by taking the LOMI size divided by the voxel size

Preliminary Collision Detection

- After obtaining the local directional depth maps, preliminary collision detection is then accomplished at every haptic rendering time *t* using *InteriorQuery*.

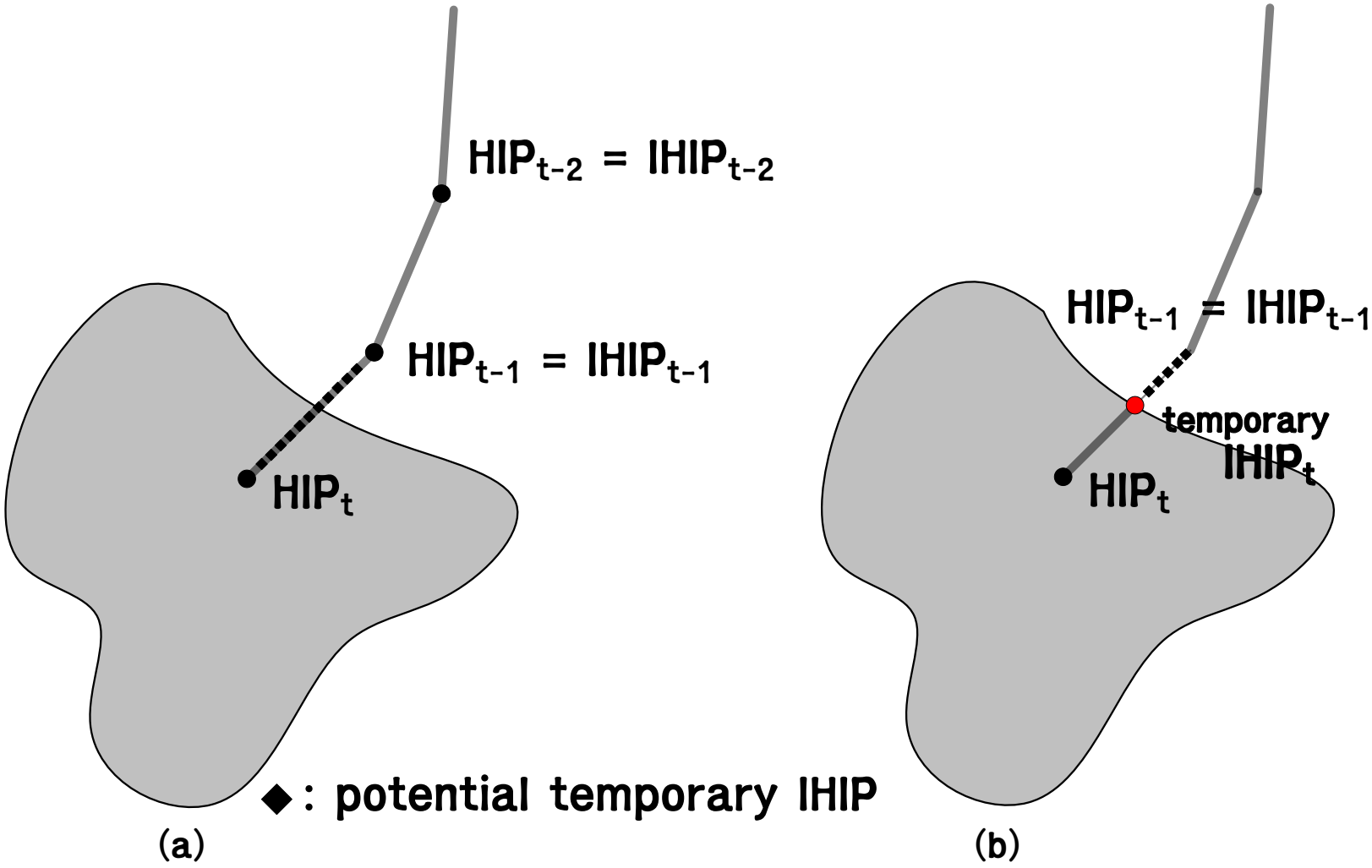
- The *InteriorQuery* function determines whether or not a given point is located inside the virtual object in the object coordinate system, by referring to depth values.

- In order to convert the depth values into position coordinates as object coordinates, *InteriorQuery* uses the following linear relation for the orthogonal projection.

$$Position = Near + (Far - Near) \times Depth$$

where *Near* and *Far* denote the near and far values of the physical clip volume of each virtual camera in the object coordinates, and *Depth* represents the normalized depth value [0, 1] acquired by each virtual camera.

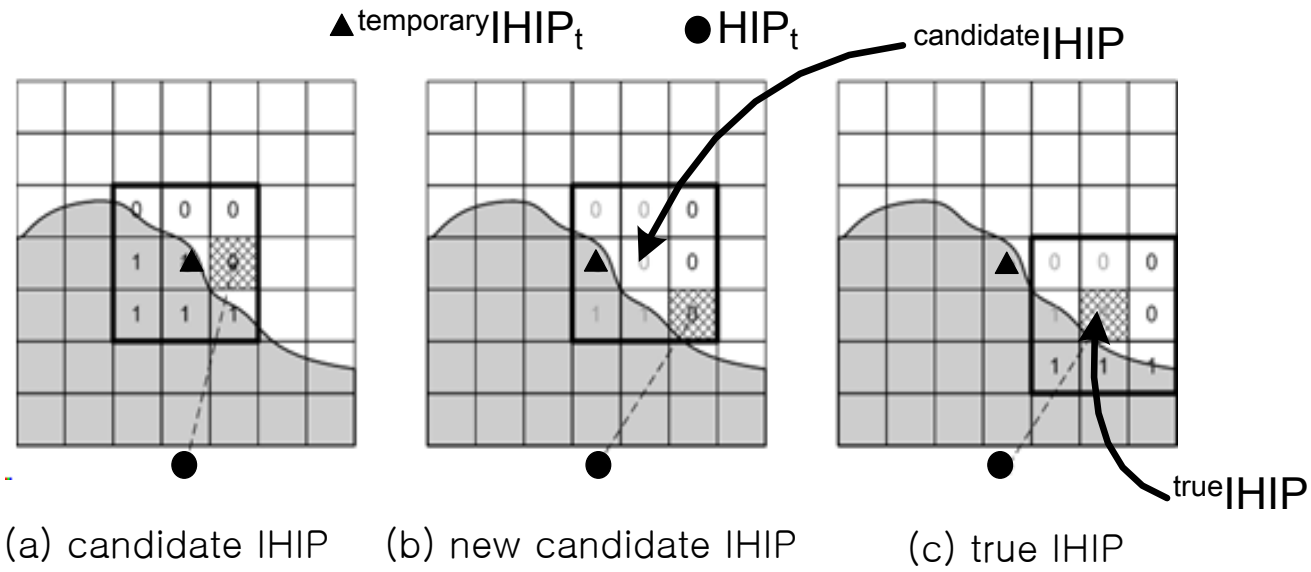
Preliminary Collision Detection



Preliminary Collision Detection

- the haptic rendering time history of the HIP, where it can be seen that the IHIP coincides with the HIP position before collision.
- divided into several points (called potential temporary IHIPs) at intervals of the desired haptic resolution.
- Each potential temporary IHIP on the line segment is then inside-checked by *InteriorQuery*.
- The first collision point from the previous $IHIP_{t-1}$ to the current HIP_t is designated as a temporary $IHIP_t$.
- this temporary $IHIP_t$ will be corrected to a true $IHIP_t$, a surface point that is the nearest the HIP_t, using an abstract LOMI.

Abstract LOMI and Nearest Neighbor Search for True IHIP

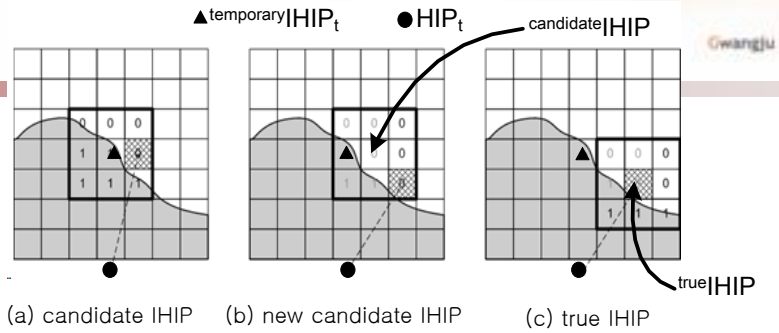


- The procedure for finding the true IHIP using an abstract LOMI

(i) Once the temporary IHIP is determined in the preliminary collision detection part, the center of the corresponding abstract LOMI is located at the temporary IHIP, the 2-dimensional case of which is shown in Fig. (a).

* the LOMI size and number of voxels in the LOMI are determined by Eq. 5 and the desired voxel size based on the desired haptic resolution.

Abstract LOMI and Nearest Neighbor Search for True IHIP



(ii) Voxel type is determined on an individual basis for the voxels neighboring the temporary IHIP by performing InteriorQuery, as shown in Fig. (a). (the voxels in a 3×3 voxmap are the neighbors surrounding the temporary IHIP).

(iii) If the voxel is an exterior voxel, the distance from the HIP is calculated until the nearest free-space voxel is selected as the candidate IHIP, without storing the voxel type. This is shown as a scratched square in Fig. (a).

(iv) For the new 3×3 voxel neighbors, shown in Fig.(b), voxel type is again individually determined by performing InteriorQuery. In this step, however, the voxels already tested in the previous step are excluded. In Fig. (a), since a candidate IHIP is selected as a voxel located just to the right of the temporary IHIP, InteriorQuery is performed only for the voxels on the right, the types of which are denoted as bold **0** in Fig.(b). Then, step (iii) is performed.

(v) If there is a voxel nearer than the current candidate IHIP, the nearer free space voxel is selected as the new candidate IHIP, as shown in Fig.(b), and step (iv) is repeated within the LOMI. Otherwise, the current candidate IHIP is selected as the final true IHIP and the search for the nearest free space voxel stops.

Abstract LOMI and Nearest Neighbor Search for True IHIP

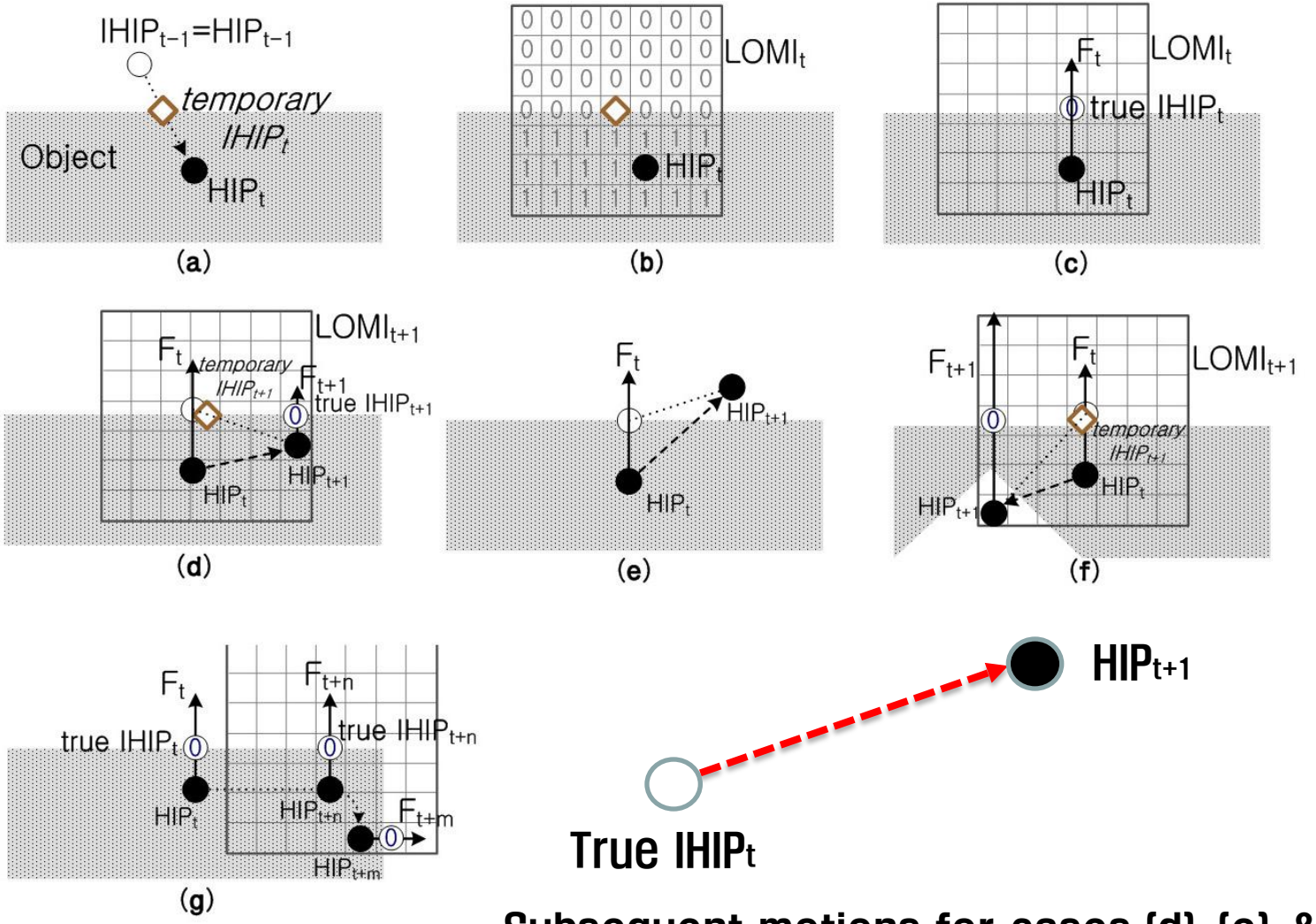
Timing Data Including Collision Detection to Find True IHIP [ms]

Desired resolution [mm]	0.7	0.6	0.5	0.4	0.3	0.2	0.1
Number of voxels in the LOMI	7^3	11^3	15^3	19^3	23^3	27^3	31^3
Physical LOMI [ms]	0.13	0.212	0.291	0.41	0.671	0.791	0.924
Abstract LOMI [ms]	0.017	0.025	0.033	0.041	0.049	0.056	0.065

- only some voxels neighboring the temporary or candidate IHIP are tested for interior and exterior types, and do not need to be stored in the physical memory in the process of finding the true IHIP.

- With this local region search in the abstract LOMI, a true IHIP location on the surface of the contacted virtual object can be efficiently determined as an order-of-magnitude increase over the previous physical LOMI.

Computation of the Normal/Friciton Collision Forces



Subsequent motions for cases (d), (e), & (f)

Computation of the Normal/Friciton Collision Forces

- Following the procedures established in procedures for finding the true IHIP using an abstract LOMI, the temporary $IHIP_t$ in the preliminary collision detection step is shown in Figure (a), the abstract $LOMI_t$ with its center located at the position of the temporary $IHIP_t$ is shown in Figure (b), and the true $IHIP_t$ within the current $LOMI_t$ is shown in Figure (c).
  - Note that types of voxels in Figure (b) are just shown to distinguish exterior and interior voxels and are not stored
  - The normal collision force is then computed based on the vector from the current HIP_t to the true $IHIP_t$.
  - The friction force is then computed based on the vector from the temporary $IHIP_t$ to the true $IHIP_t$

- The next movement of HIP_t can be classified into three cases:

Case-1) stay inside the object for objects with thick parts (thicker than LOMI)

Case-2) move out to the same free space as HIP_{t-1}

Case-3) move out to the free space opposite HIP_{t-1} for objects with a very thin part

Computation of the Normal/Friciton Collision Forces

- preliminary collision is again performed using an inside-outside check for the points in the line segment from the true $IHIP_t$ to HIP_{t+1} to determine a Temporary $IHIP_{t+1}$.

Case-1 : If HIP_{t+1} moves to a new inside position

- In this case, the first collision occurs at the same free space as HIP_{t-1} while the last collision at the interior.
- To find the true $IHIP_{t+1}$, the procedure can be repeated for the new abstract $LOMI_{t+1}$ at the temporary $IHIP_{t+1}$, and the nearest free-space voxel to HIP_{t+1} is then selected as the true $IHIP_{t+1}$
- The normal collision force at time *t+1* is computed based on the vector from the current HIP_{t+1} to the true $IHIP_{t+1}$.
- The friction force at time *t+1* is computed based on the vector from the temporary HIP_{t+1} to the true $IHIP_{t+1}$.

Case-2: If HIP_{t+1} is located at the same free space as HIP_{t-1}

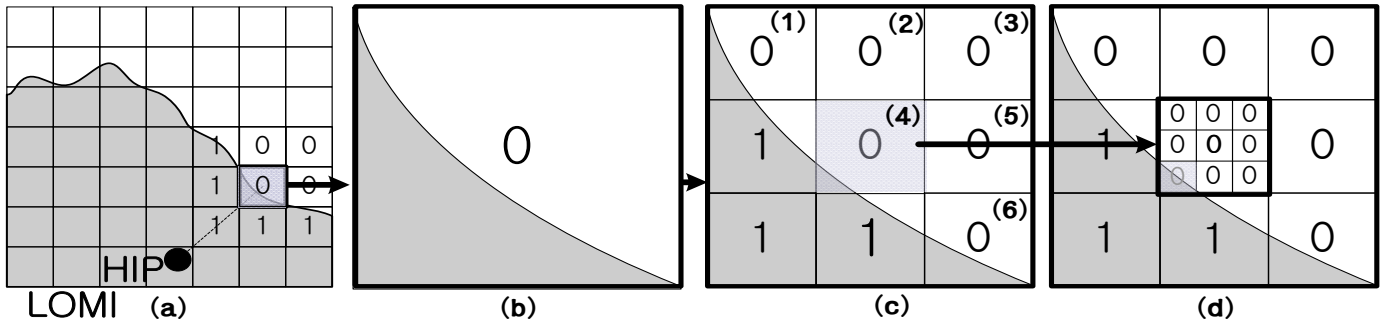
- a preliminary collision test shows no collision.
- no contact forces are generated.

Computation of the Normal/Friciton Collision Forces

Case-3 : If HIP_{t+1} moves to the free space opposite HIP_{t-1}

- for objects with a very thin part, the true $IHIP_{t+1}$ can be located at the opposite side free space.
- In this case, the first collision occurs at the same free space as HIP_{t-1} while the last collision at the free surface at the opposite side.
- In this case, the same true IHIP finding procedure with the new abstract $LOMI_{t+1}$.
- The normal collision force at time *t+1* is computed based on the vector from the current HIP_{t+1} to the true $IHIP_{t+1}$.
- The friction force at time *t+1* is computed based on the vector from the temporary HIP_{t+1} to the true $IHIP_{t+1}$.

Accuracy Enhancement (Smoothness) with Sub-voxelized LOMI



- In deciding the IHIP, the IHIP resolution should be the same as the voxel size.
  - the limited resolution of the IHIP may lead to a sensation of surface roughness, even if the local surface of a virtual object is very smooth.
  - For the object geometry that is approximately modeled using a small number of triangular meshes with the Phong shading technique so that it looks like a smooth surface, though the object may not feel smooth when the user scans the surface because the proposed LOMI-based algorithm does not have information for the surface normal vectors.
  - a sub-voxelization algorithm can be applied to improve the resolution of the IHIP and thus smoothness of the smooth surface (Q: how do we know smoothness of the given surface without knowing a priori ???)

Accuracy Enhancement (Smoothness) with Sub-voxelized LOMI

(i) When the true IHIP on the exterior voxel is determined as shown in Figure (a), only the voxel containing the IHIP is sub-voxelized, as shown in Figure (c).

(ii) This voxel may be divided into three regions in a 2D plane, as in spatial partitioning methods

(iii) then, the newly partitioned voxel types are determined by utilizing the bilinearly interpolated depth data for the sub-voxels.

(iv) If a new IHIP voxel is determined, then it can further be divided into lower level sub-voxels, as shown in Figure (d).

(v) This process can be repeated until the size of the sub-voxel meets the desired IHIP resolution (smoothness) of an object, while considering haptic rendering time budget.

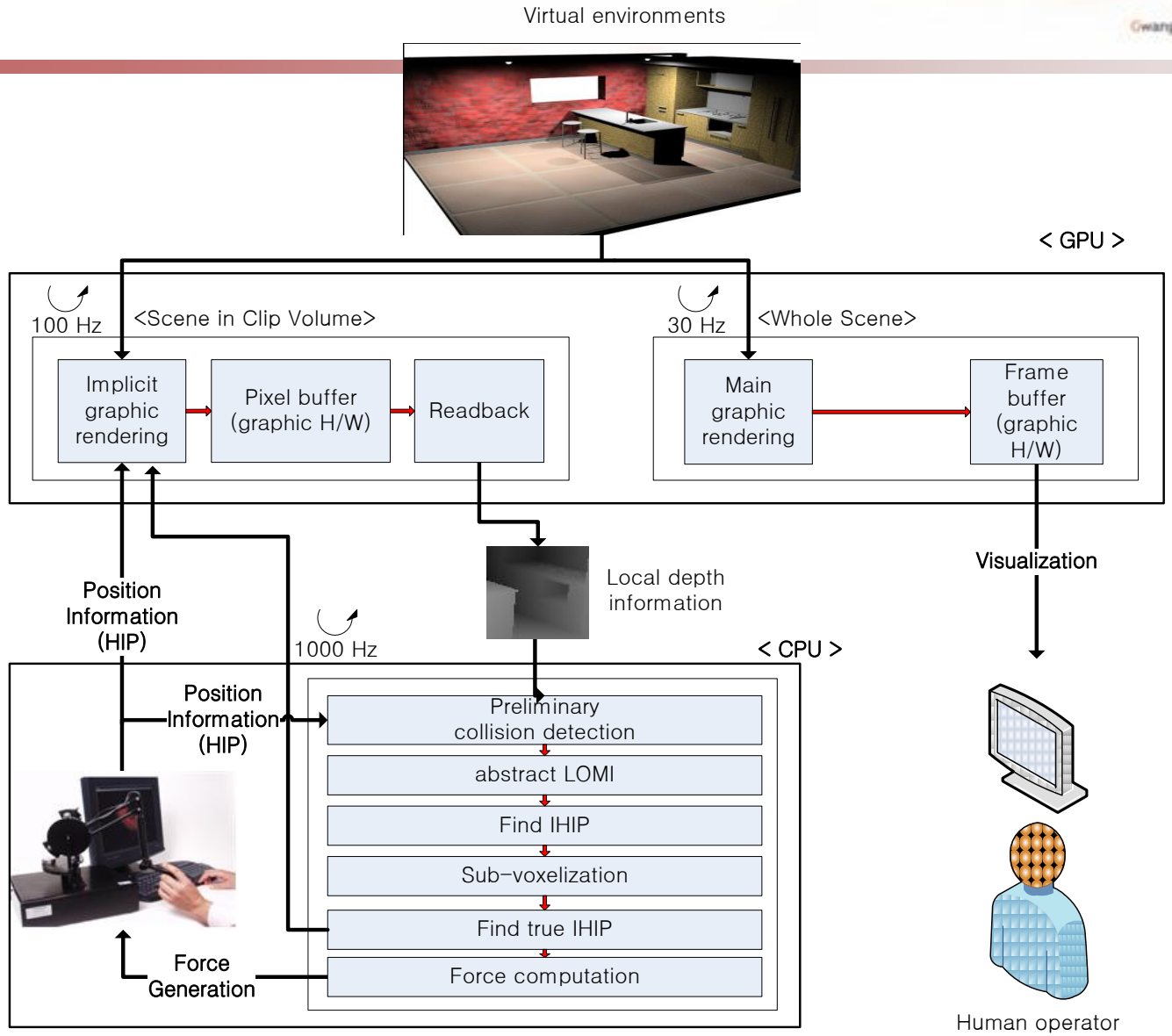
Accuracy Enhancement (Smootheness) with Sub-voxelized LOMI

Timing Data with Sub-voxelization [ms]

No. of iterations for sub-voxelization	1	2	3	4	5	6	7
IHIP resolution [mm]	$2/3^1$ (0.67)	$2/3^2$ (0.22)	$2/3^3$ (0.07)	$2/3^4$ (0.02)	$2/3^5$ (0.01)	$2/3^6$	$2/3^7$
Time for finding IHIP [ms]	0.011	0.019	0.028	0.036	0.044	0.052	0.060

- the time cost for sub-voxelization is very small
- shows an almost linear increase in time cost with respect to the number of sub-voxelization iterations.
- 0.01 mm is good for smooth surface feeling.
- 0.02 mm is in the order of haptic device position resolution

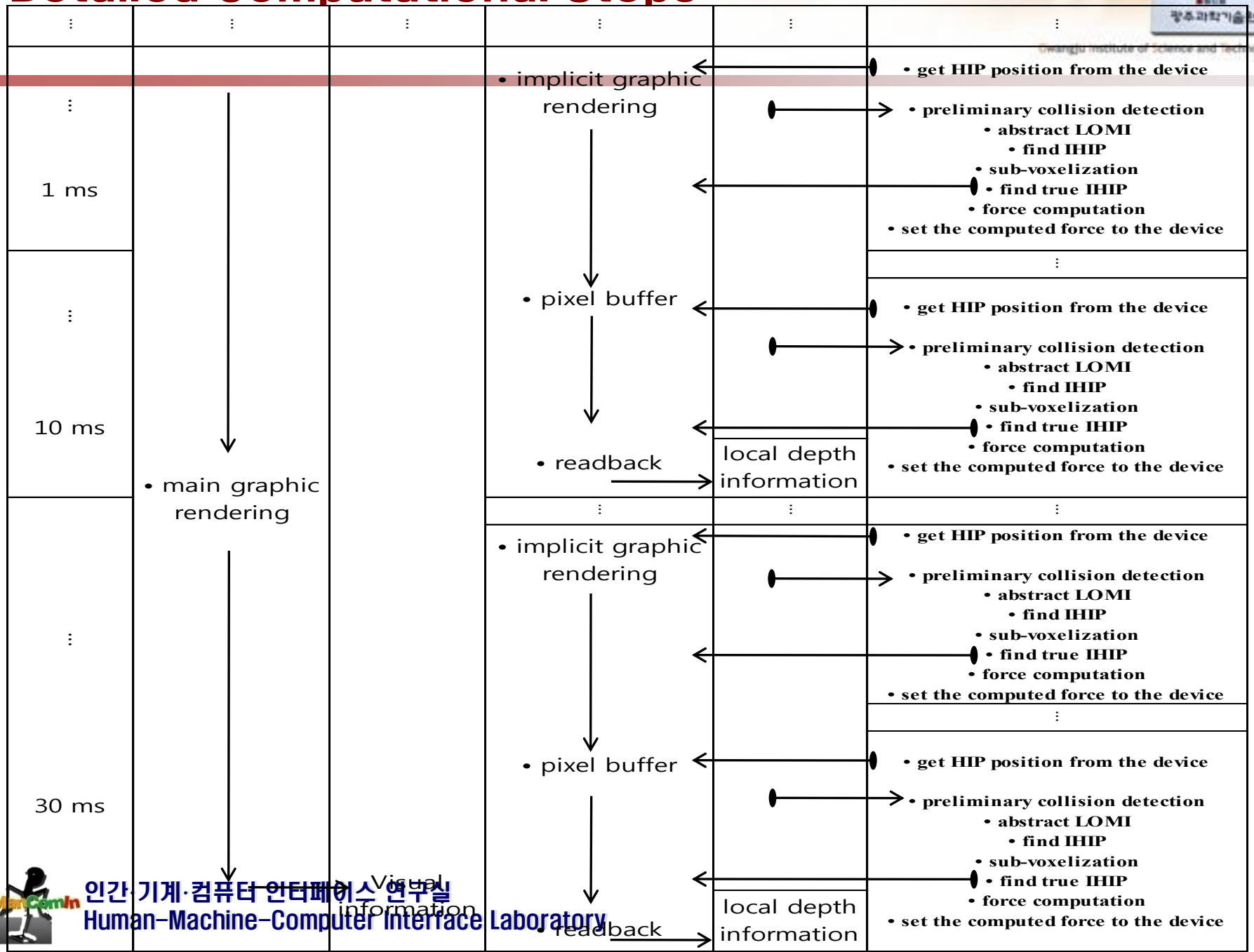
overall data flow among processors and devices



overall data flow among processors and devices

- The proposed hybrid CPU/GPU and LOMI-based haptic rendering algorithm includes many computational steps
- The performance of each step should be investigated to ensure the optimum performance.
- The overall data flow among processors and devices
- the position data (HIP) is obtained from the haptic device and is fed to the CPU at the haptic rendering rate to update the LOMI and subsequently compute the collision response force.
- This data along with the computed true IHIP data is then fed to the additional graphic rendering pipeline in the form of a transformation matrix for checking Cases-I, II, III for computing normal and friction forces.

Detailed Computational Steps



Detailed Computational Steps

- The time schedule of the proposed LOMI algorithm for sampling rates of 30 Hz, 100 Hz, and 1000 Hz for the main graphic rendering, the implicit additional graphic rendering, and the haptic rendering, respectively.

- The haptic and the additional graphic rendering loops are asynchronous, haptic rendering may refer to depth data that is not completely updated at the additional graphic rendering.

- This lack of data can potentially cause problems in terms of maintaining stability of the proposed algorithm.

  ➔ use a double buffering method that enables the haptic rendering to refer to the new, complete, and self-consistent depth data.

Algorithm Evaluation and Optimization

- The computational time in this algorithm can be separated into three groups

(1) Six implicit additional graphic rendering times for obtaining the directional depth information, the performance of which depends mainly on the graphic rendering method.

(2) Readback time to store depth values in the main memory, the performance of which depends on both the number of pixels in the implicit window and the GPU bus structure.

(3) Time for haptic rendering based on the abstract LOMI with sub-voxelization, the performance of which depends mainly on the number of voxels in the abstract LOMI.

Implicit Rendering Evaluation and Optimization

- Indexed geometry allows vectors to be shared by sevral triangles
  - Puts an array of indices into the array of vertices, and each set of three indices determines a triangle
- Display list is a group of graphics language commands and arguments that has been stored for subsequent execution
  - a convenient and efficient way to name and organize a set of OpenGL commands.
  - The graphics language may be instructed to process a particular display list by providing a number that uniquely specifies it.
- Vertex buffer object (VBO) stores vertex array data to video memory to make rendering faster, similar to the display list, thereby yielding a higher update rate.
  - VBO differs from the display list in the sense that the display list cannot be dynamically modified and once the display list is compiled, the user may have to generate a new display list, and delete the old one, so that new data can be sent to the video memory.

Implicit Rendering Evaluation and Optimization

Timing Data for Implicit Graphic Rendering [ms]

Rendering Method	Number of meshes in 3D model							
	0.2M	0.4M	0.6M	0.8M	1.0M	1.2M	1.4M	1.6M
Indexed Geometry	66	117	184	234	296	351	402	465
Display List	0.07	0.12	0.18	0.25	0.31	0.34	0.43	0.46
VBO	0.07	0.13	0.18	0.23	0.29	0.33	0.38	0.48

- The presented rendering time is measured based on off-screen rendering (for additional graphic rendering) in which only geometry information is processed, with no processing of color information.

- For a comprehensive evaluation, time measurements are conducted by changing the number of meshes in the virtual model.

Implicit Rendering Evaluation and Optimization

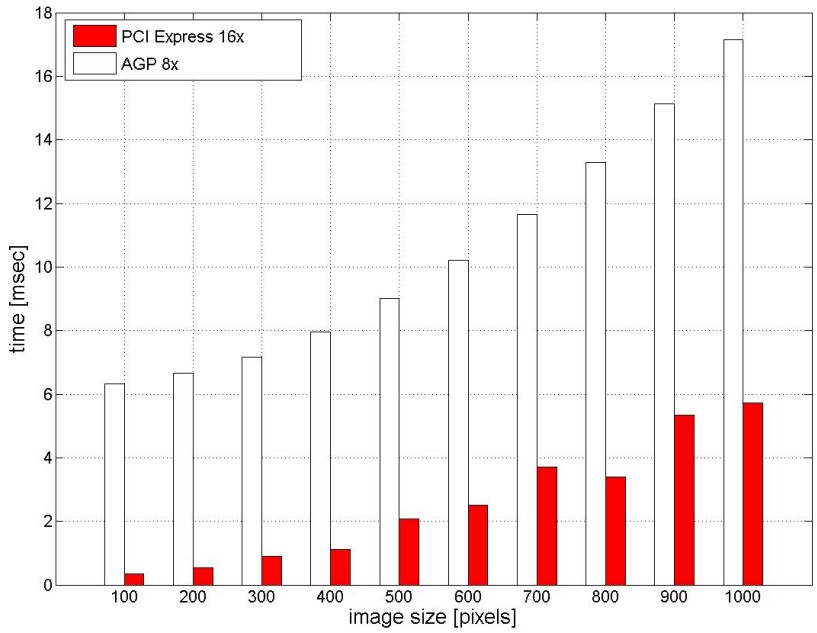
- The measured times in Table 1 represent the average value of one thousand measurements.

- The indexed geometry rendering method displays an almost linear increase in rendering time as the object complexity increases.

- Conversely, both the display list and VBO show almost constant times regardless of the geometry complexity.

- Therefore, VBO was selected for the additional implicit graphic rendering because display lists cannot dynamically modify the object geometry information.

GPU Readback Evaluation and Optimization



- The time needed to read back the depth buffer from the graphics hardware to the main memory.
- Two types of commercial graphics hardware were compared in terms of readback time [ms], including: AGP (NVIDIA GeForce FX 5900) and PCI-E (NVIDIA Quadro Fx 4400).

- The readback time is measured for an implicit window and for a changing number of pixels, from (100×100) to (1000×1000).
- The measured time again represents the average value of one thousand measurements.

GPU Readback Evaluation and Optimization

- The figure shows that the PCI-E graphics hardware is far more efficient than the AGP graphics hardware, as the readback time of PCI-E for (1000×1000) pixels is less than 6 ms.

- The implicit window size of (400×400), which requires about 1 ms for readback, is sufficient for accurate and efficient rendering.
  - confirmed that the entire additional graphic rendering performance was not adversely affected by the readback process.

- For larger numbers of pixels requiring more than a 1-ms readback time, a more powerful GPU and increased bus system bandwidth is required.
  - the processing power of GPUs and the bandwidth of bus systems have rapidly improved as a result of recent demands from the gaming industry.

Haptic Rendering Evaluation and Optimization

- The computational time, including the time needed for preliminary collision detection, finding the true IHIP through an abstract LOMI, and sub-voxelization.
  - run on a Pentium IV computer having a 2.8GHz CPU and 1GB of RAM for haptic rendering, and a GeForce 6800 Ultra graphics card with a PCI-Express bus structure for graphic rendering.

- To measure the computational time, the time needed for collision detection and finding the true IHIP using either the physical or abstract LOMI was first measured with respect to the desired haptic resolution (voxel size) and voxel number in the abstract LOMI.
  - For a simple comparison based on typical cases, an abstract LOMI size of 2 mm was chosen and the desired haptic resolution was varied from 0.7 mm to 0.1 mm.

Haptic Rendering Evaluation and Optimization

Timing Data Including Collision Detection to Find True IHIP [ms]

Desired resolution [mm]	0.7	0.6	0.5	0.4	0.3	0.2	0.1
Number of voxels in the LOMI	7^3	11^3	15^3	19^3	23^3	27^3	31^3
Physical LOMI [ms]	0.13	0.212	0.291	0.41	0.671	0.791	0.924
Abstract LOMI [ms]	0.017	0.025	0.033	0.041	0.049	0.056	0.065

- The computation from preliminary collision detection to finding the true IHIP with the abstract LOMI takes an order of magnitude less time than with the physical LOMI.

- The efficiency and accuracy of finding an IHIP through sub-voxelization process: time measurements were conducted for the 0.2 mm voxel size and up to 0.1 μm resolution (0.2 mm/3^7 ≈ 0.1 μm; 7 iterations of the sub-voxelization process for a 0.2-mm voxel), the voxel to be sub-voxelized was divided into 27 voxels at each iteration.

Haptic Rendering Evaluation and Optimization

Timing Data with Sub-voxelization [ms]

No. of iterations for sub-voxelization	1	2	3	4	5	6	7
IHIP resolution [mm]	$2/3^1$	$2/3^2$	$2/3^3$	$2/3^4$	$2/3^5$	$2/3^6$	$2/3^7$
Time for finding IHIP [ms]	0.011	0.019	0.028	0.036	0.044	0.052	0.060

- The time cost for sub-voxelization is very small (in the order of 0.01 ms)
- it linearly increases with the number of iterations of subvoxelization

* at each sub-voxelization only 27 voxels are sought, thereby increasing the speed and accuracy for finding an IHIP.

- The proposed sub-voxelization can provide accurate IHIP resolution by partitioning the selected voxel.

Demonstrative Examples

- present both implemented examples of the proposed LOMI-based haptic rendering algorithm and the results of benchmark tests that include the haptic rendering time.

- As benchmark tests, both OpenHaptics API and the CHAI3D haptic library (CHAI3D, 2005) were selected to compare the haptic rendering times with the proposed algorithm

- The proposed algorithm was implemented on a PHANToM Premium 1.5 haptic interface (Phantom haptic device, 2003) without using the OpenHaptics haptic rendering algorithm
  - The computations were run on a Pentium IV with 2.8GHz CPU and 1GB of RAM for haptic rendering, and with a GeForce 6800 Ultra graphics card with a PCI-Express bus structure for graphic rendering.

Demonstrative Examples

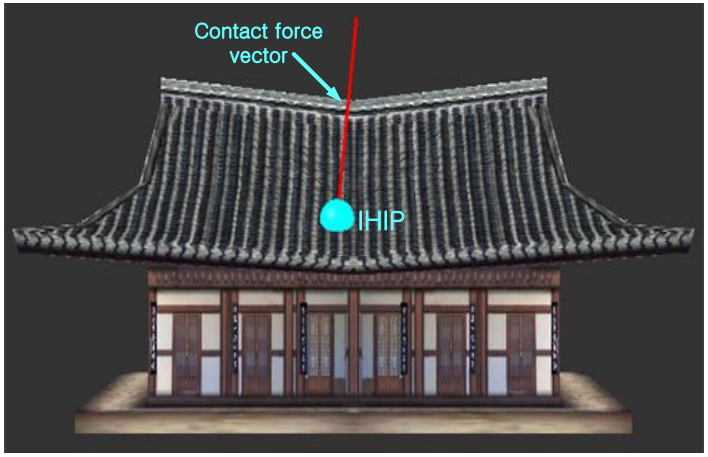
- The application was multi-threaded, with the haptic force computation thread running at high priority at a 1 kHz.
  - To display the VEs, graphic rendering was run at a rate of 30 Hz.

- The size of the abstract LOMI was selected to be 2 mm based on the assumption that the maximum moving speed of the haptic device and the haptic rendering rate were 1 m/s and 1 kHz, respectively.
  - The desired haptic resolution was set at 0.2 mm

- thus, the abstract LOMI had voxels of size (11×11×11) and (100×100) pixel size was used for implicit graphic rendering.

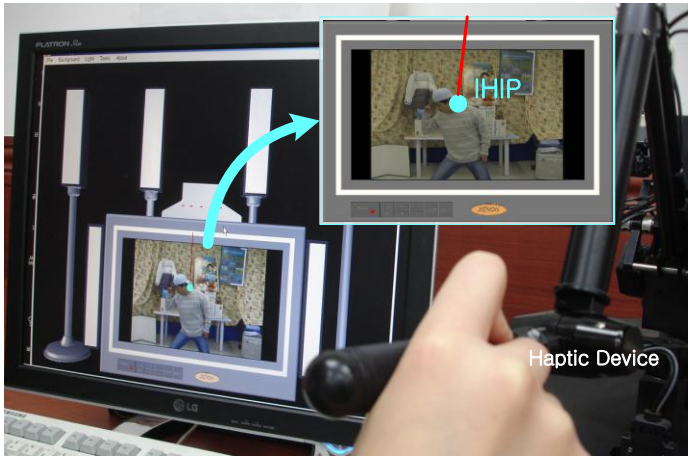
- These conditions could generate a very precise IHIP resolution (sub-voxelized voxel size) of about 2 micro-meter—adequate for local surface details of the chosen
  demonstration examples as well as for real-time voxelization and memory occupation.

Demonstrative Examples

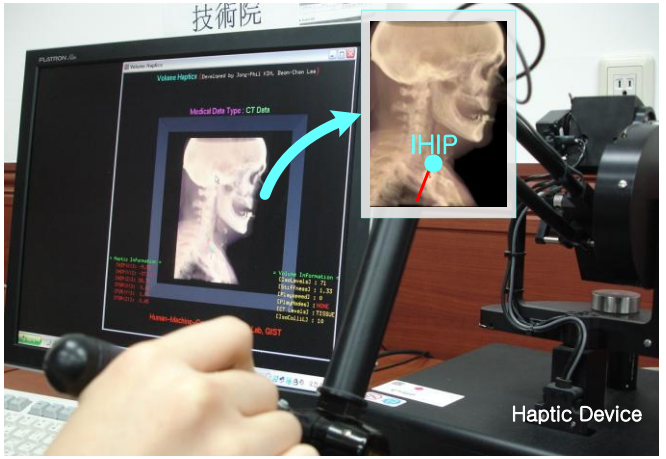


(a) Simple object (100K polygons)

(b) Complex environment (1,000K polygons)



(c) Hybrid virtual environment
(2.5D depth data, 300K 3D polygons)



(d) Medical volume data
(256×256×256 CT data)

Demonstrative Examples

- Haptic rendering algorithm with six virtual cameras was then applied to various relatively simple [Figure (a)], highly complex [Figure (b)], hybrid [Figure (c)], and voxel data [Figure (d)].

- Figure (a) shows their haptic interaction with a relatively simple static object composed of 100K triangular meshes.
  - In the figure, the green point is the IHIP position and the red line is the contact force vector that indicates the direction and magnitude of the force calculated between the HIP and IHIP.

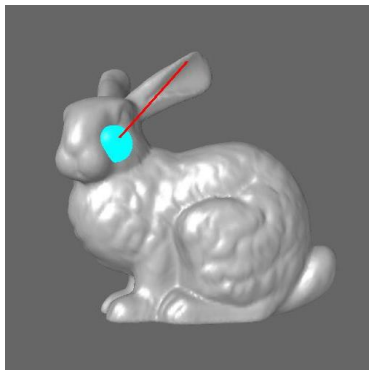
- Figure (b) shows the haptic interaction with a highly complex 3D VE, where the number of polygons is about 1000K.
  - In this large-scale VE, a user can navigate using a keyboard and can adjust the scale of the VE using a mouse, which can provide a full haptic exploration for the VE.
  - The figure show that even such a highly complex VE can be accurately and efficiently interacted with in real-time using the proposed algorithm.

Demonstrative Examples

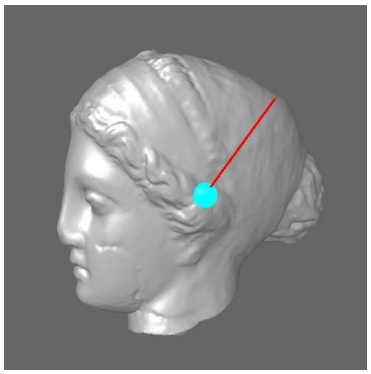
- Since the proposed haptic rendering algorithm is independent of the data representation type, the proposed algorithm can also be applied to interact with hybrid VEs, as shown in Figure 13(c), which contains a 2.5D depth image (height-field data) captured by Z-Cam™ (Z-Cam™, 2006) and a conventional 3D model (300K polygonal model).
  - This example demonstrates that the proposed algorithm enables haptic interactions with different types of datasets without requiring any additional effort, such as data unification or a complicated data hierarchy.

- Figure 13(d) shows haptic interactions with voxel data from the CT of a human body with no further triangulation or spatial partitioning.

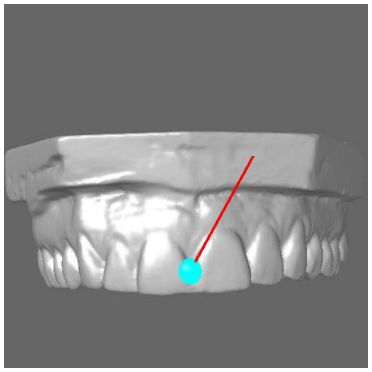
Benchmark Tests



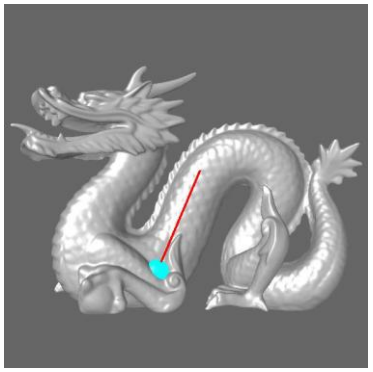
(a) Bunny (69,451 polygons)



(b) Venus (134,356 polygons)



(c) Teeth (233,204 polygons)



(d) Dragon (871,414 polygons)

- In order to show the advantages of the proposed algorithm over existing algorithms
  - tested 3D models with simple (about 70K: Bunny), medium (about 134K: Venus; about 233K: Teeth), and complex (about 870K: Dragon)
  - polygons based on the following algorithms
    - proposed haptic rendering algorithm
    - OpenHaptics API
    - CHAI3D haptic library

Benchmark Tests

Haptic Rendering Time [ms]

	Number of vertices	Number of polygons	Proposed algorithm	OpenHaptics	Chai3D
Bunny	35,947	69,451	0.060	4.08	22.25
Venus	67,108	134,356	0.061	5.31	41.58
Teeth	116,604	233,204	0.061	22.42	143.37
Dragon	437,645	871,414	0.062	40.03	430.39

- Only the haptic rendering time, including the time for collision detection, locating the IHIP, and force computation, was measured for each haptic rendering algorithm.
- Table does not include the rendering time for the six virtual cameras
- The measured time represents the average value of 1000 measurements.
- Haptic rendering time is about 0.06ms, independent of the complexity of the 3D model, clearly demonstrating that the algorithm is very fast

Benchmark Tests

- In the case of OpenHaptics, the haptic rendering time increases with the complexity of the 3D model as it uses a feedback buffer that stores points, line segments, and polygons when finding IHIPs as well as to detect collisions.
  - Thus, even for a simple model (about 70K: Bunny) the haptic rendering time of OpenHaptics was about 4.08 ms, which is too slow for 1-kHz haptic rendering.

- Haptic rendering time (more than 22 ms) of CHAI3D shows unacceptable performance for all 3D models.
  - This poor performance is because CHAI3D uses a hierarchical data structure for the object's geometry, such as axis-aligned bounding boxes, to detect collisions and to find IHIPs.
  - Searching such a data structure for collision detection and for finding IHIPs requires most of the available haptic rendering time

Conclusions, Discussion, and Future Works

- Then, to increase the process speed for finding the IHIP, the abstract LOMI with the nearest neighbor search algorithm was proposed, and to enhance the resolution of the IHIP, a sub-voxelization process was developed for the selected candidate voxel.

- Nevertheless, improvement in determining the IHIP position was demonstrated by use of the proposed sub-voxelization using bilinear interpolated depth data, which significantly increased the IHIP resolution.
  - for the cases where the surface geometry has narrow tube-like shapes, parts of which cannot be detected with even six cameras, OpenHaptics using a feedback buffer would likely outperform the proposed algorithm in terms of accuracy.

Conclusions, Discussion, and Future Works

- Fair performance comparison in terms of timing data or analytical algorithms, pitting our work against other non-open source works seems to be very difficult. However, the novel features of our work can be summarized as follows.

  - There is no need to have prior knowledge of data hierarchy.

  - scalable to object data representations (enables haptic interactions for any kind of object that is visually rendered without requiring different haptic rendering algorithms for different object data representations).

  - enables a fast haptic servo rate that can be maintained regardless of scene complexity or size.

Conclusions, Discussion, and Future Works

- cannot detect collisions between the HIP and the surface when this is invisible, e.g. the inside surface of a calabash, because the cameras are attached on the sides of a haptic workspace that encloses the outer surface of the object.
  - only the visible parts can be touched from the proposed "from-outside-looking-inside" approach.
  - For these cases, a "from-inside-looking-outside" algorithm will be investigated in a future study.

- works include the potential for the proposed algorithm to be extended to dynamically changing 3D VEs or to deformable bodies since the proposed algorithm does not need any complex data hierarchy for representing geometric data.

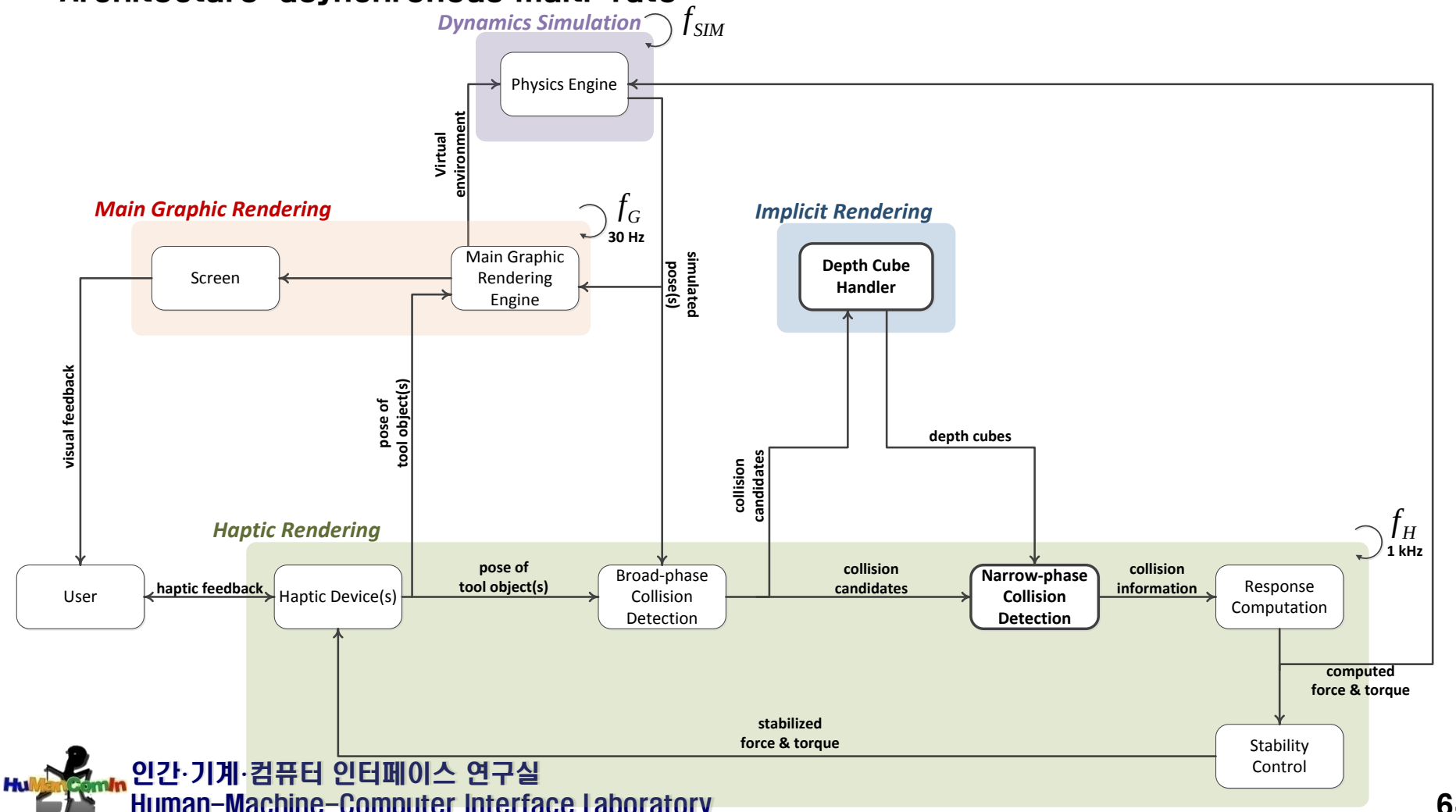
Conclusions, Discussion, and Future Works

- Furthermore, even though the current investigation focused on a 3-DOF haptic interaction mode, the basic concept and algorithm can be extended to full 6-DOF haptic interaction by considering the location of several probe points in a rigid 3D interaction tool

Extention to 6DOF Depth image-based haptic rendering

Algorithm Overview

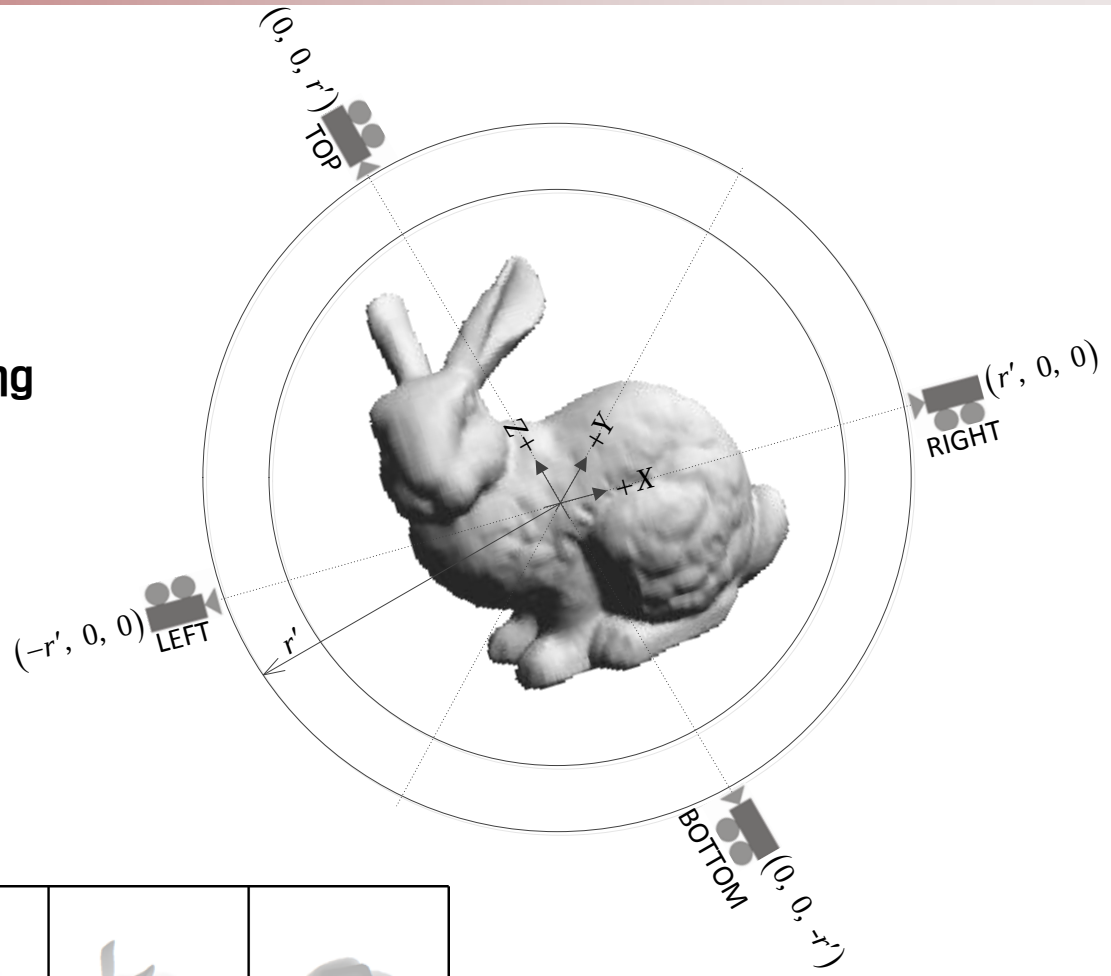
- Architecture: asynchronous multi-rate

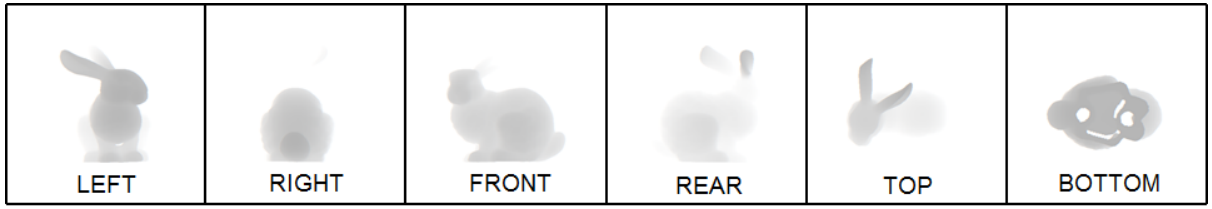


6DOF Depth image-based haptic rendering

- **Depth Cube Structure**

- represents object geometry by using six sided orthogonal depth images
- easy and fast point-in-object test





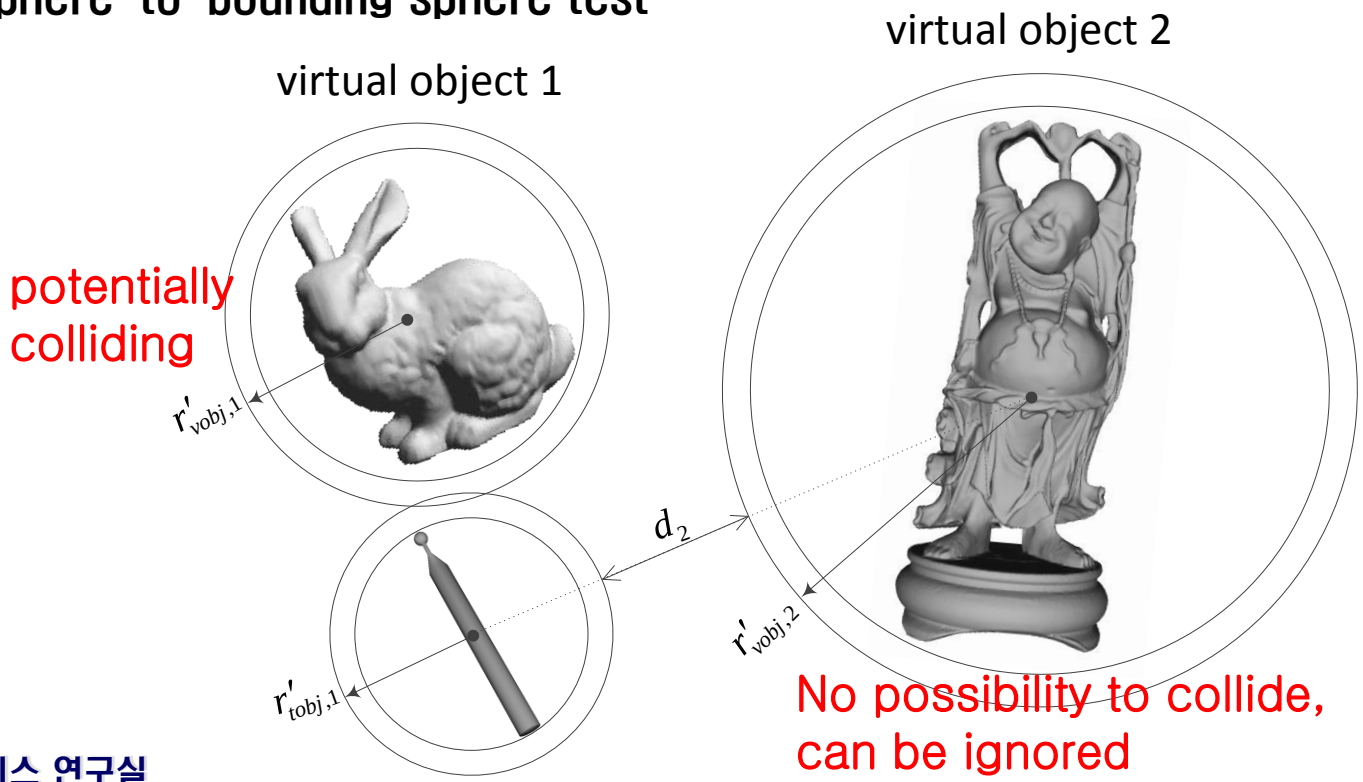
6DOF Depth image-based haptic rendering

- Collision Detection

- two phases: broad-phase & narrow-phase

- Broad-phase collision test

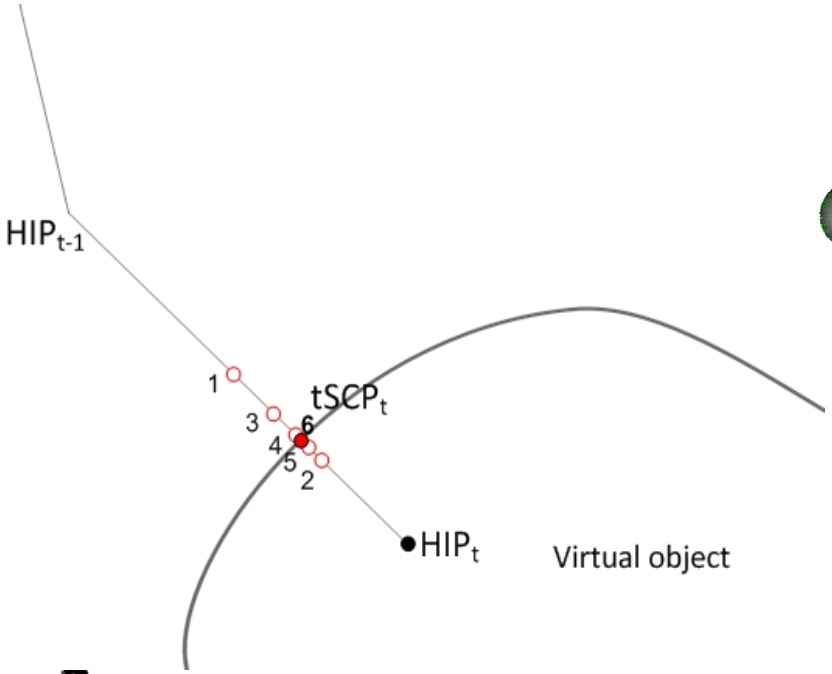
- uses simple bounding sphere-to-bounding sphere test

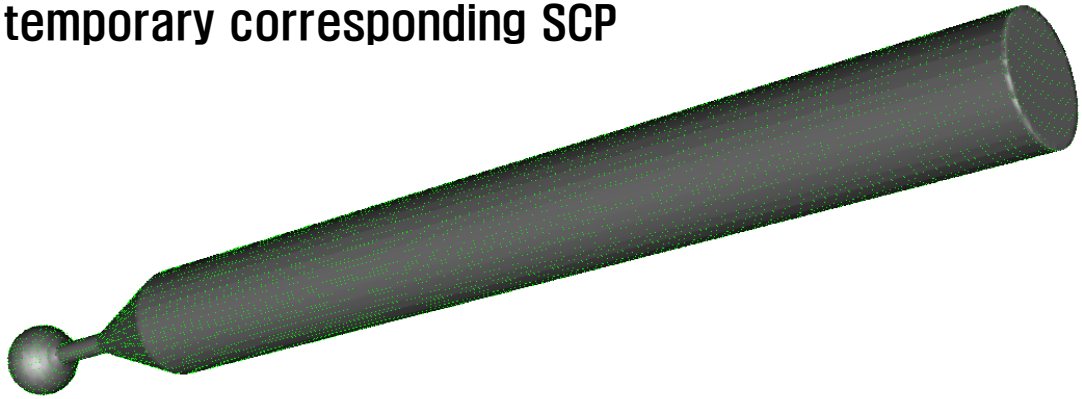


6DOF Depth image-based haptic rendering

- Narrow-phase collision test

- performs exact collision detection
- For each sampled point, find a temporary SCP
  1) check if a current sampled point is in contact,
  2) search the accurate location of the temporary corresponding SCP
  3) Compute the penetration depth



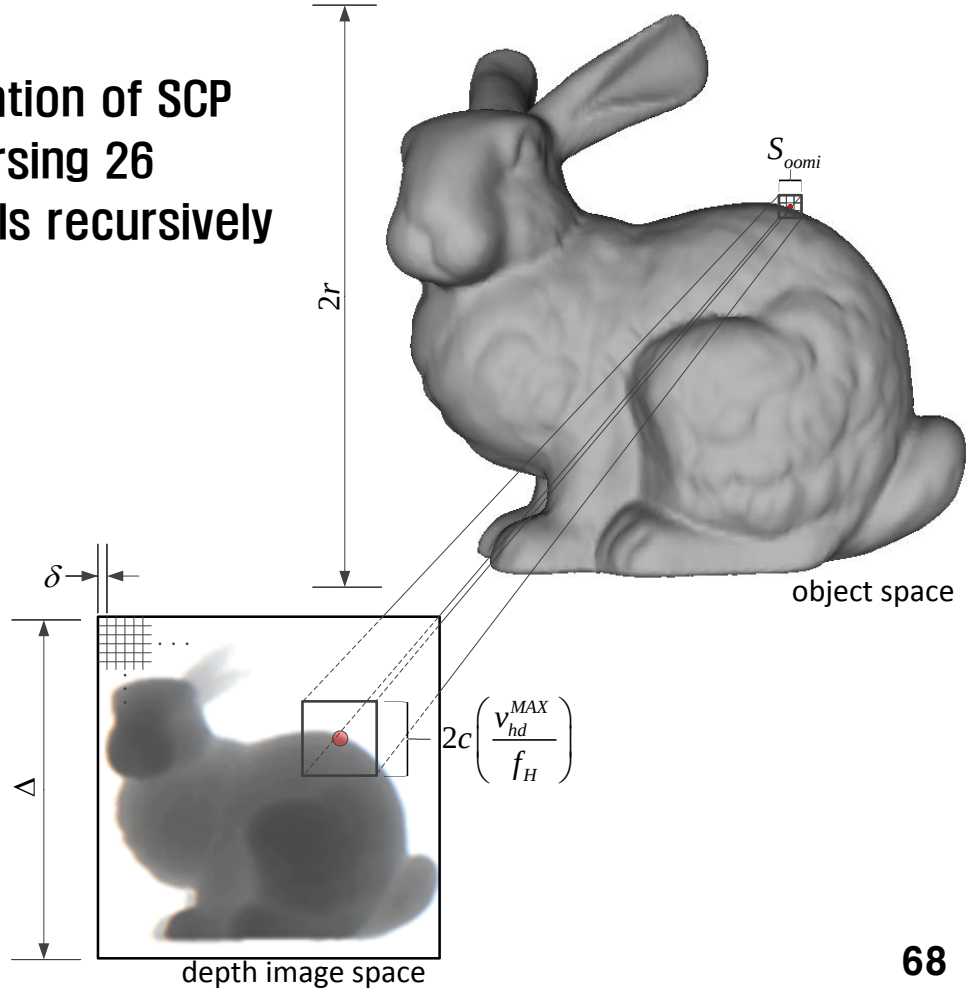


The tool object is represented as a collection of sampled points

6DOF Depth image-based haptic rendering

- Narrow-phase collision test
  - OOMI structure is created after the temporary SCP is found
  - OOMI structure: a spatiotemporal 3D filtering window

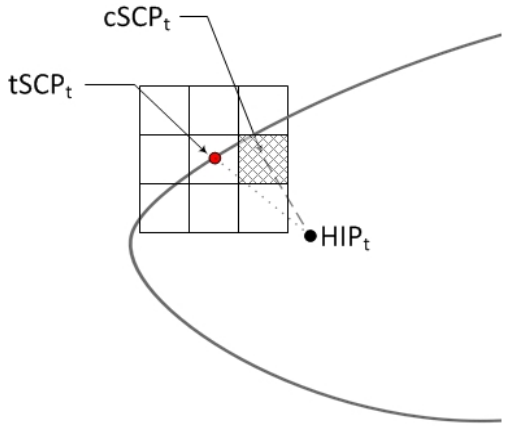
OOMI contributes to single out the ideal location of SCP (true SCP) near the temporary SCP, by traversing 26 neighbor cells and their sub-partitioned cells recursively

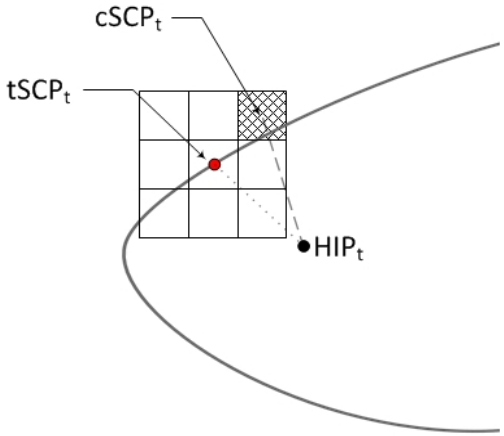


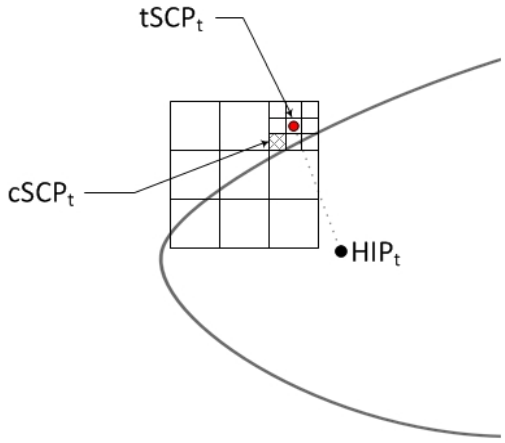
6DOF Depth image-based haptic rendering

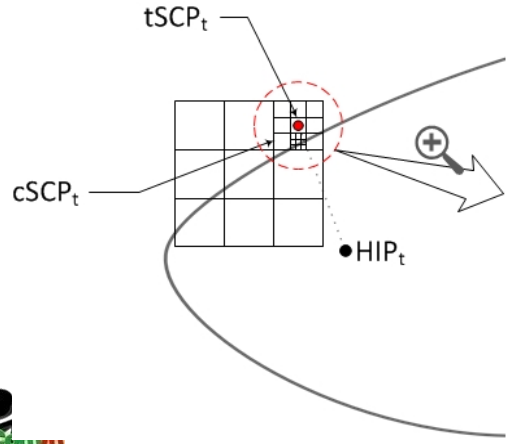
- Narrow-phase collision test

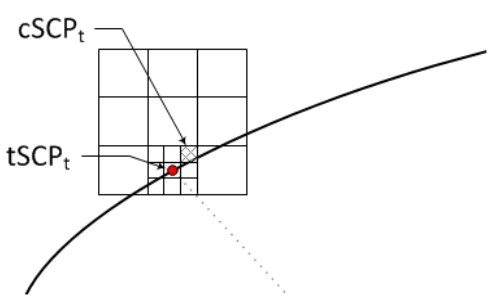
– Finding the true SCP by using OOMI structure

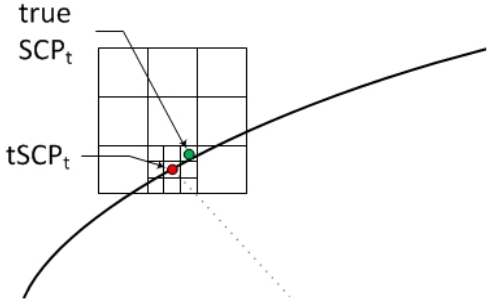












6DOF Depth image-based haptic rendering

Performance

- Due to the inherent feature of the image-based approach, the performance is not ruled by the complexity of virtual object

Experiments	number of vertices	number of triangles	haptic rendering time in CPU (in average)	haptic rendering time in CPU (in average)
			Where sampled points are 350	Where sampled points are 700
Bunny	71,894	138,902	0.488 msec	1.112 msec
Happy Buddha	1,087,304	2,175,432	0.651 msec	1.386 msec
Dragon	1,132,196	2,265,660	0.660 msec	1.344 msec

Discussion and Future Works

Discussion

- Often fail to see the details of the geometry of virtual object, when the object has a kind of cavity that cannot be successfully captured by the depth camera
- Unless a sufficient depth image resolution is provided, the proposed algorithm may experience force discontinuity
- The performance degrades when the number of the sampled point of the tool object becomes large

Future works

- Intelligent/adaptive selection of surface feature points of the tool object
- Apply to deformable bodies and fluids

Thank You for your attention

ANY QUESTIONS ?