Deferred Splatting

Gaël GUENNEBAUD
Loïc BARTHE
Mathias PAULIN

IRIT – UPS – CNRS
TOULOUSE – FRANCE

http://www.irit.fr/~Gael.Guennebaud

Plan

- Complex Scenes: Triangles or Points ?
- High-Quality Splatting: Really efficient ?

- Deferred Splatting:
  - Accurate point selection
  - Temporal Coherency
- Applications: Occlusion culling & SPT

- Results
- Future Works

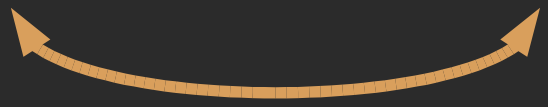
Motivations

- Real-Time rendering of Complex Scenes
- Triangles:
  fully supported by graphics HW, but...
  - tiny triangles are inefficient
  - multi-resolution can be very tedious
- One solution is Points:
  - no connectivity, no texture map, no...
  - ➔ multi-resolution rendering: simple & efficient
  - but...

Motivations

- One solution is Points, but ...
  - Large magnification: low quality
  - Flat surfaces: inefficiency
    - ➔ hybrid, triangles and points are complementary:

      use triangles when points become less efficient



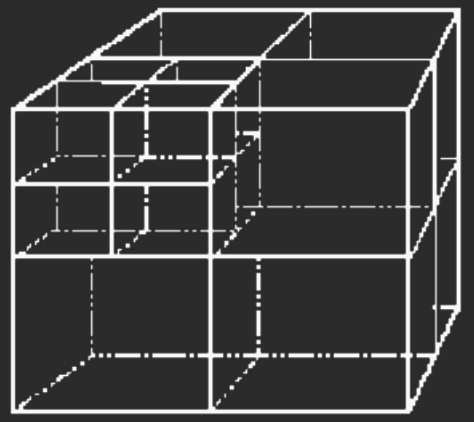
  - High-Quality point rendering is expensive
    - ➔ deferred splatting !

Efficient Point Rendering

- 2 issues:
- How to select points that have to be rendered ?
- How to render the points ?

Efficient Point Rendering

- How to select points that have to be rendered?
  - Store points into a hierarchical data structure
    (kd-tree, octree, hierarchy of bounded spheres, ...)
  - Recursive traversal with
    - visibility culling
      (view frustum,back-face,occlusion, ...)
    - LOD selection
      (local density estimation, remove superfluous points)



- *How to render the points ?*

Efficient Point Rendering

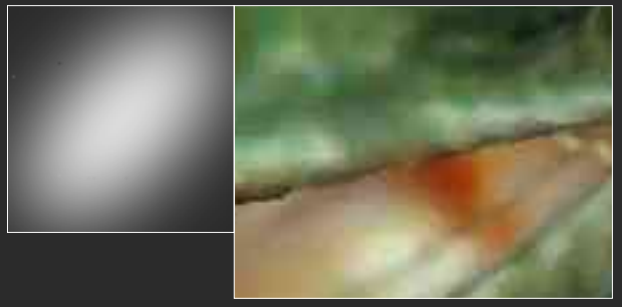
- *How to select points that have to be rendered?*
- How to render the points ?
  - Efficiency =>
    - graphics HW
    - splatting approach

GPU Point Rendering
quality & performance issues


Standard GL_POINTS

(render a disk instead of
a square is almost free)


Opaque ellipses

High-Quality Splatting

(accumulation of elliptic Gaussian,
e.g. EWA Surface Splatting)

60–85	35–45	6–10

Number of million of points per second
(GeForceFX 5900 under Linux)

*vs **44 M of small triangles** per second*

Complex Scenes : Example

- Scene ~ 6800 trees
- 1 tree ~ 750k points

➔ 5000 Millions points

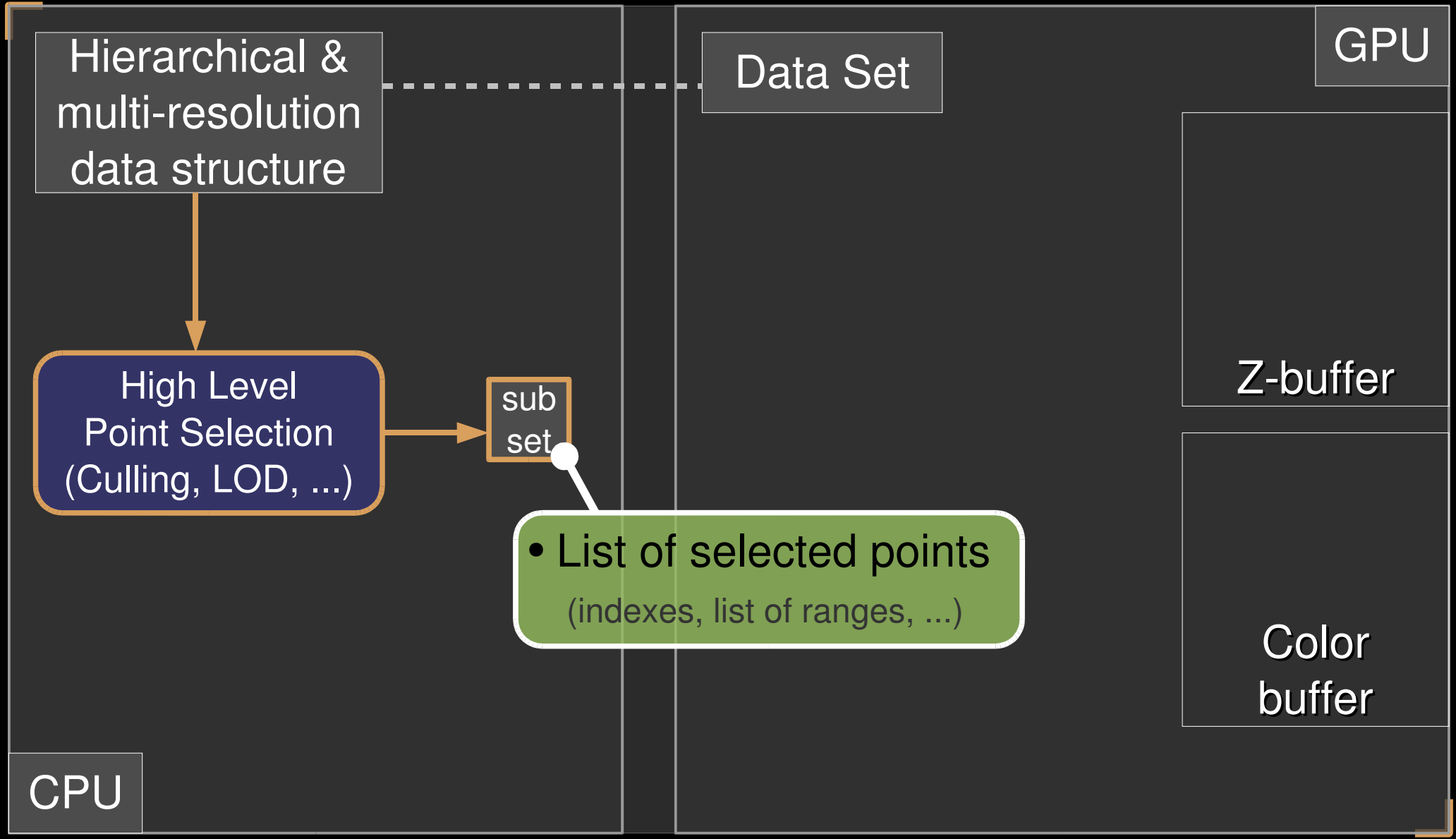
- After High-Level culling & LOD: ~ 4 M points are still potentially visible and have to be rendered
- But in fact only 150k are really visible !

Our Solution : Deferred Splatting

- is similar to “deferred shading”:
  - Defer expensive rendering computations to visible points only
- is based on:
  - An accurate point selection
  - Temporal coherency

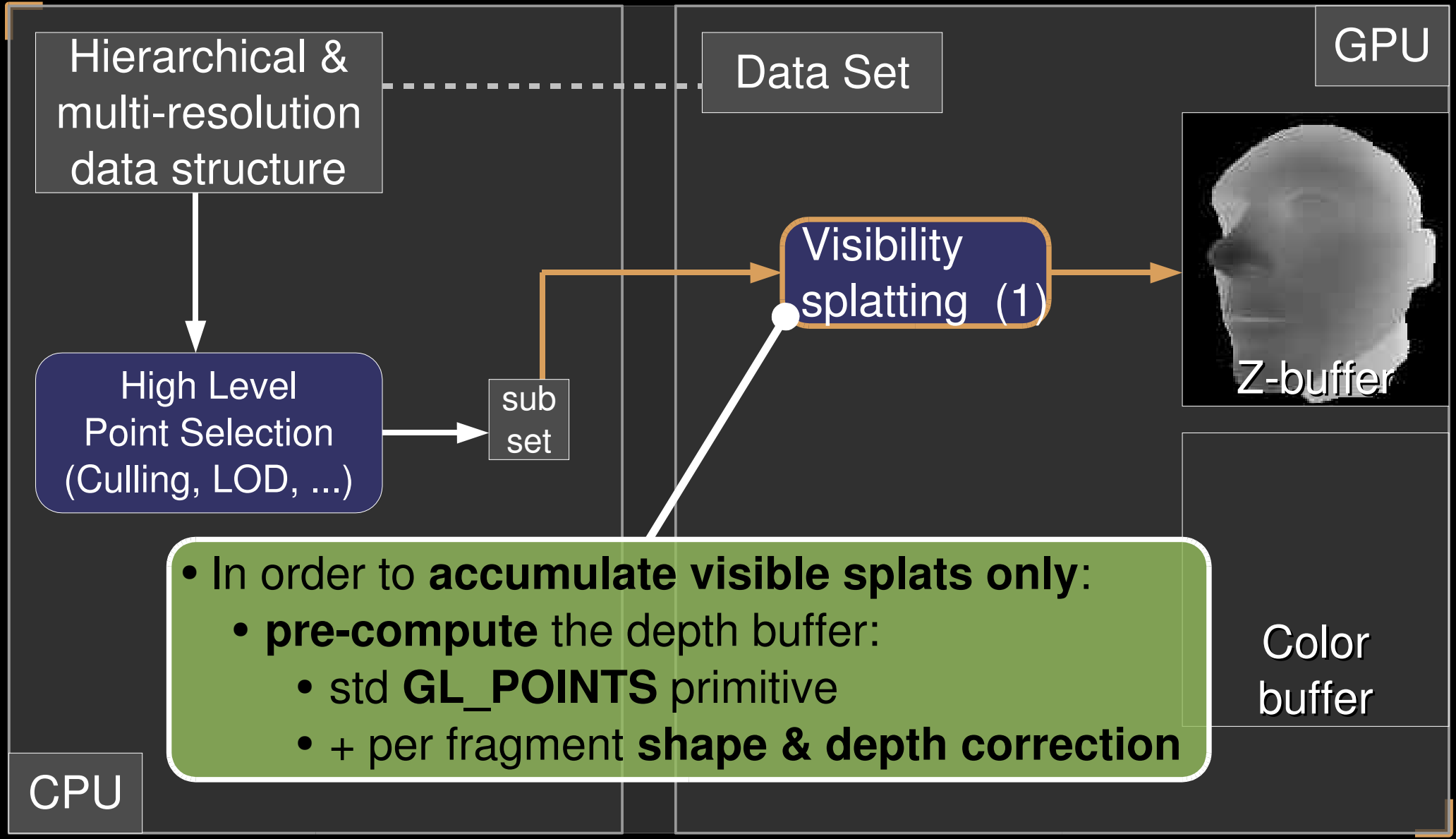
High-Quality Splatting on GPU

a multi-pass algorithm



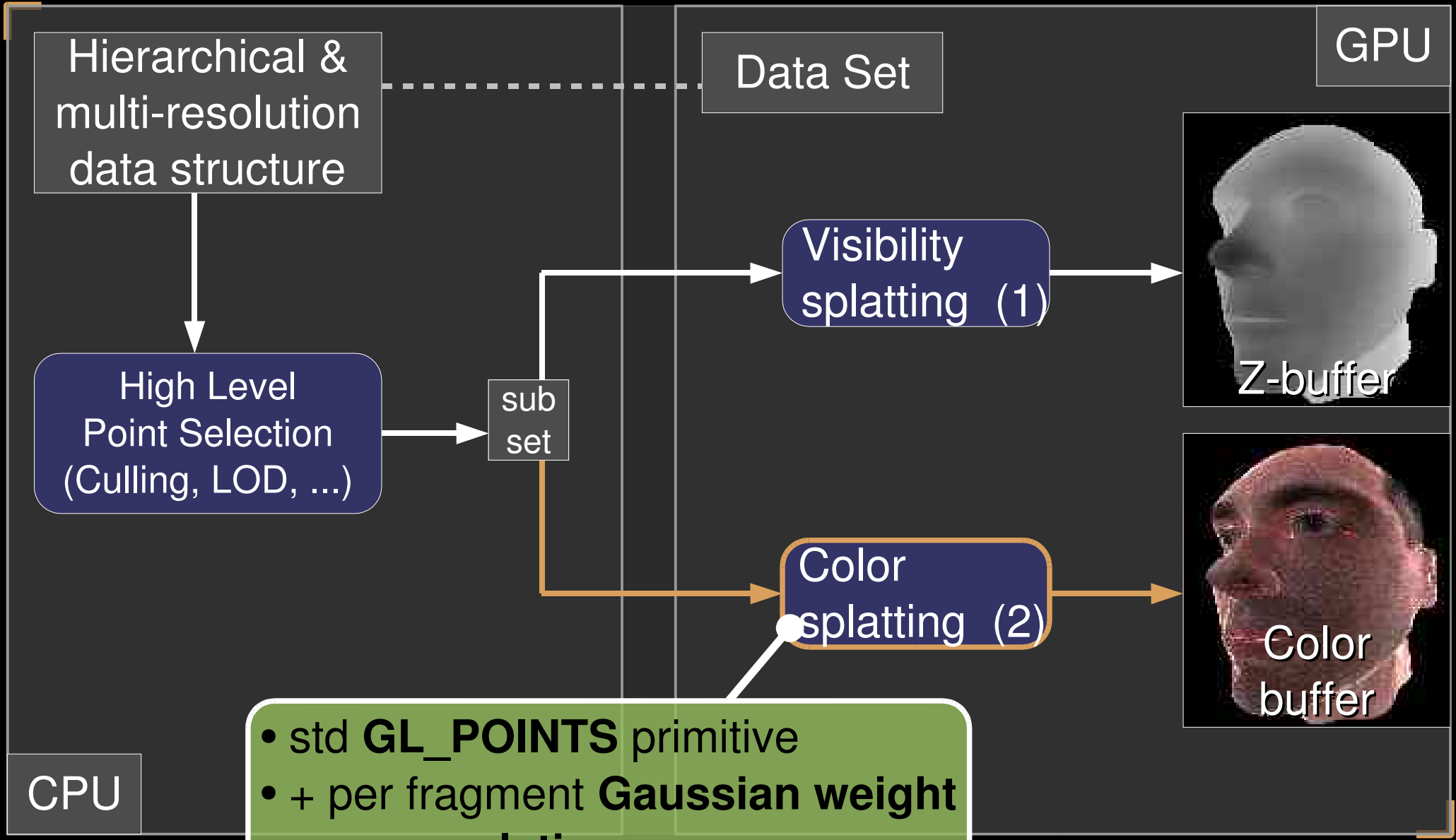
High-Quality Splatting on GPU

a multi-pass algorithm



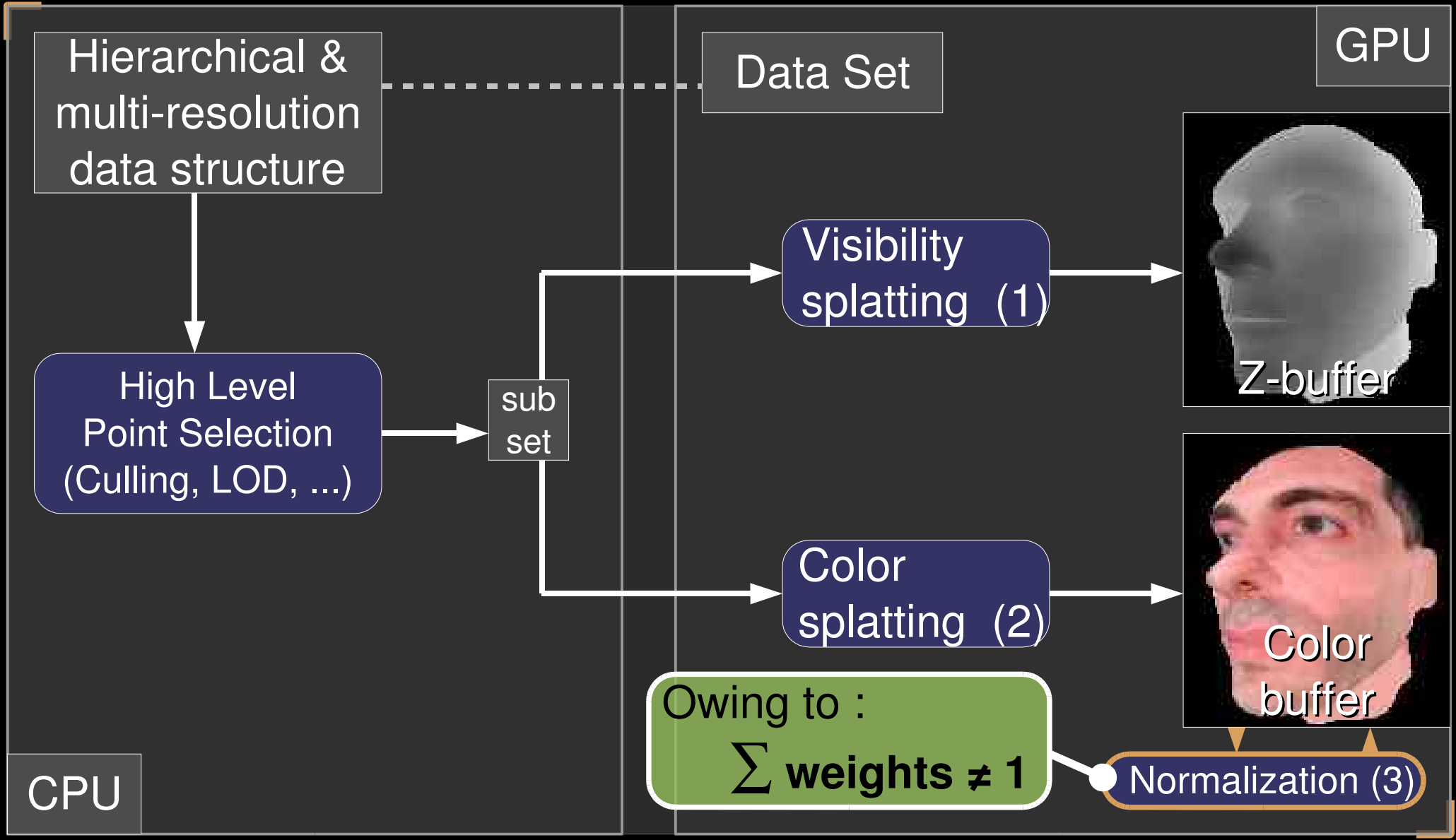
High-Quality Splatting on GPU

a multi-pass algorithm



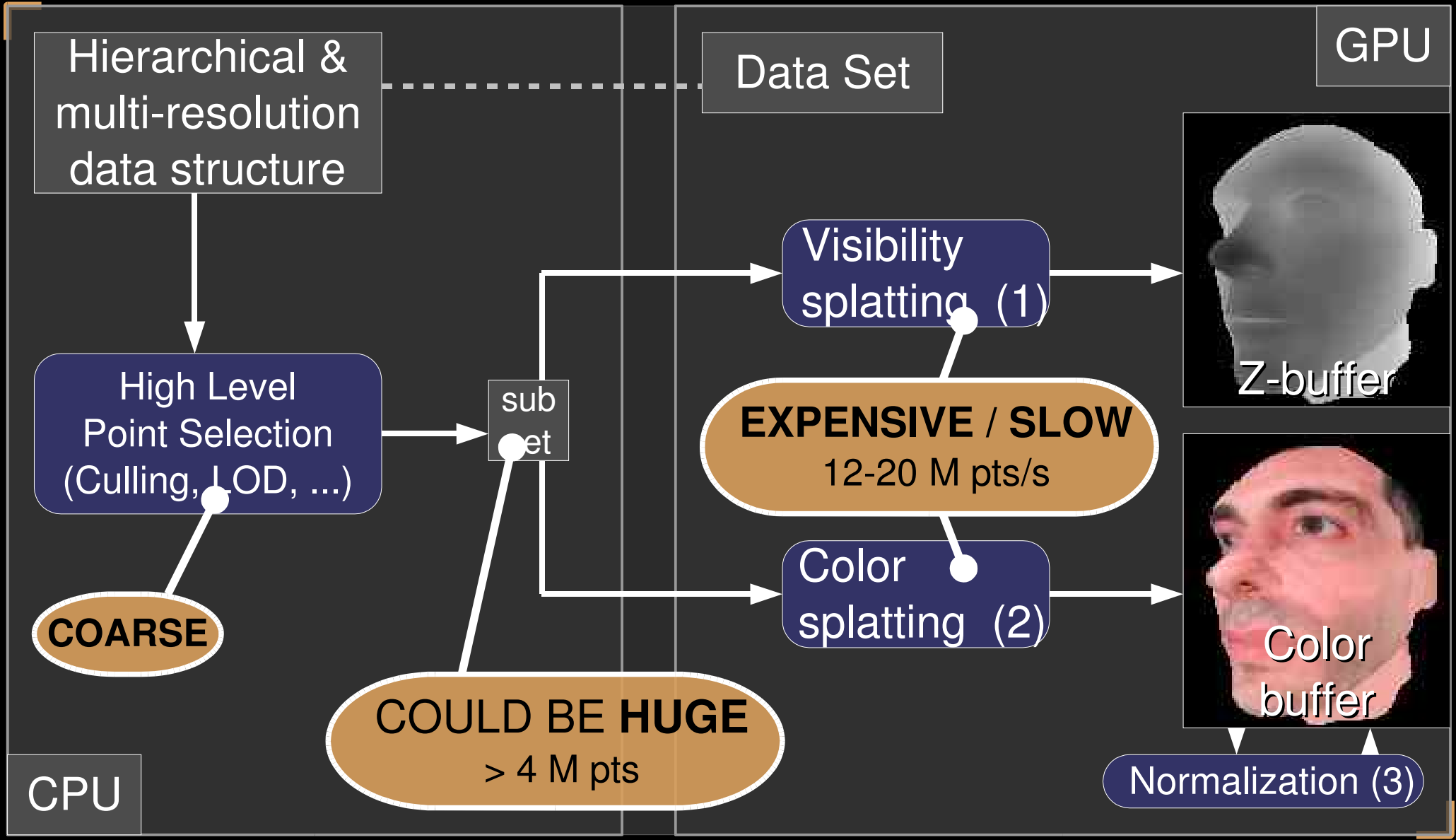
High-Quality Splatting on GPU

a multi-pass algorithm



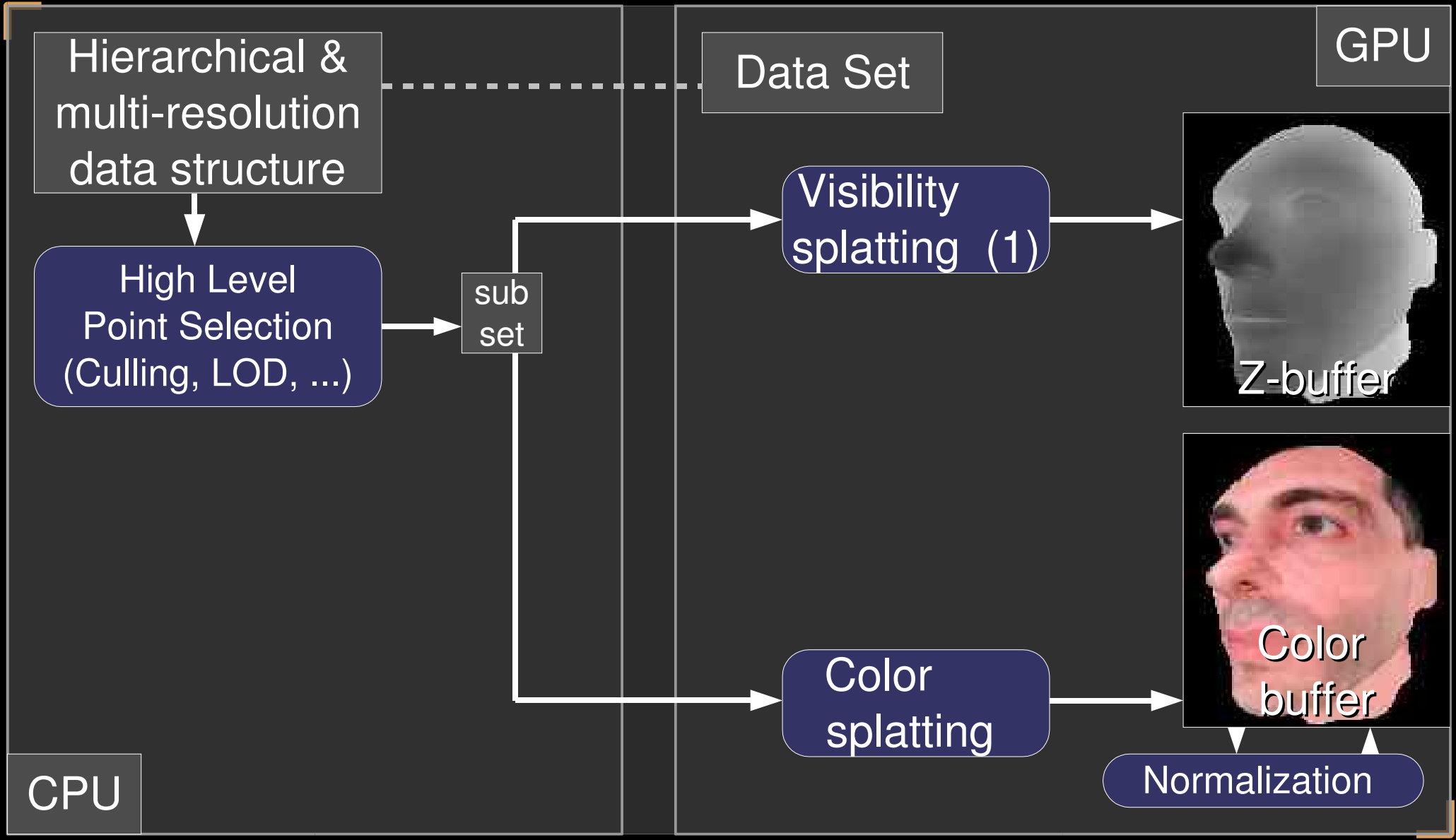
High-Quality Splatting on GPU

[analyse]

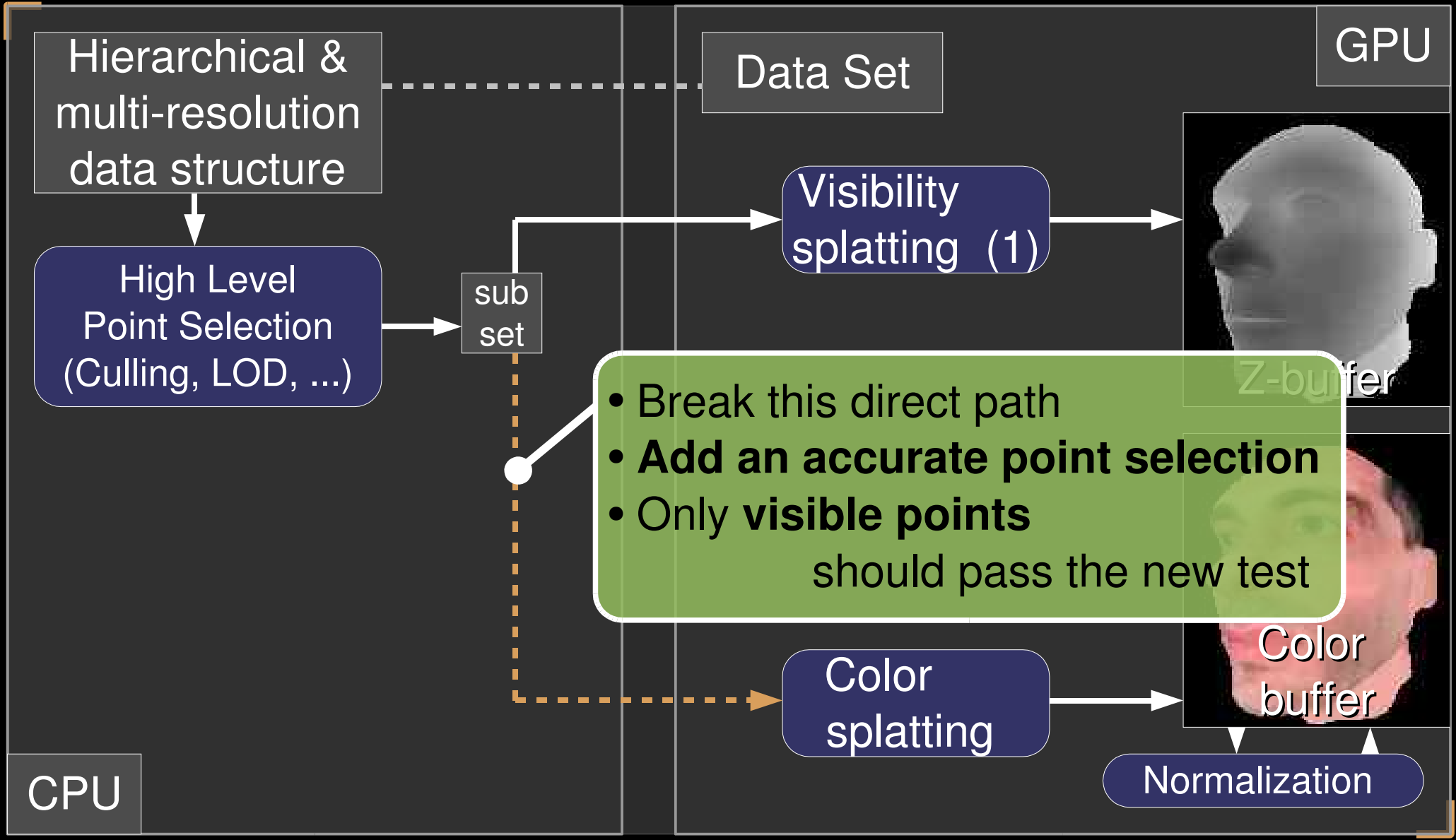


The Deferred Splatting Algorithm

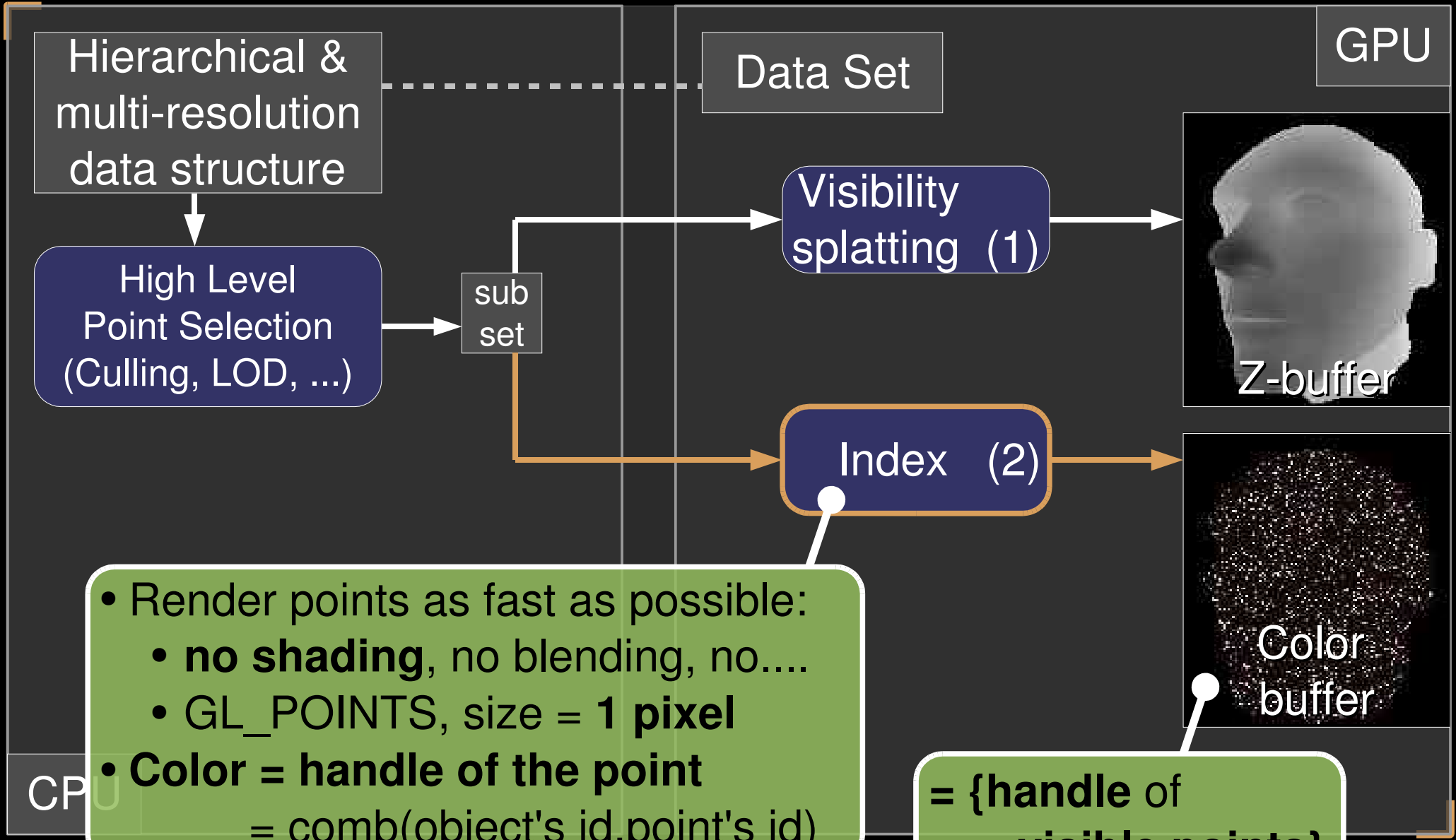
Accurate Point Selection



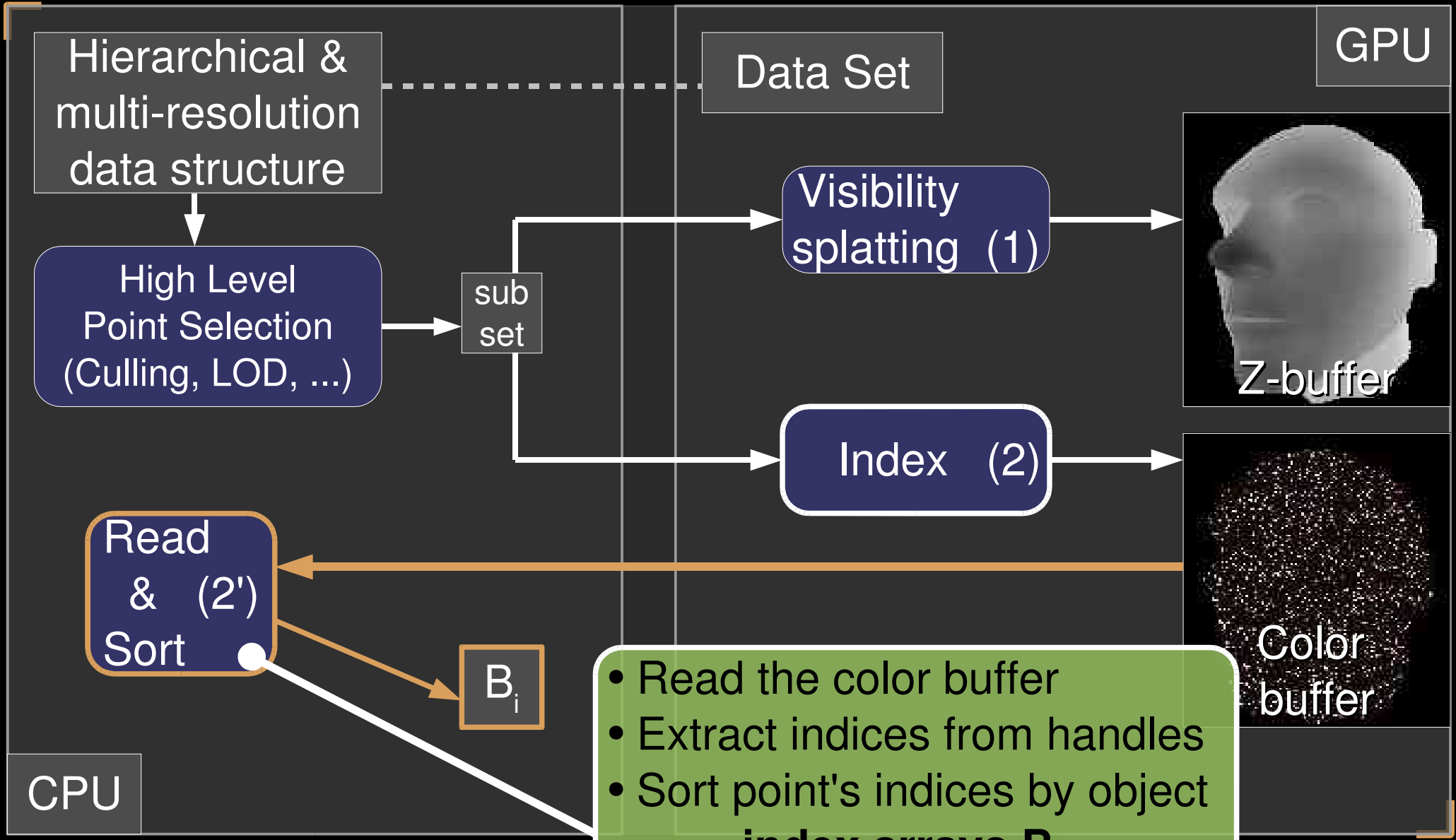
Accurate Point Selection



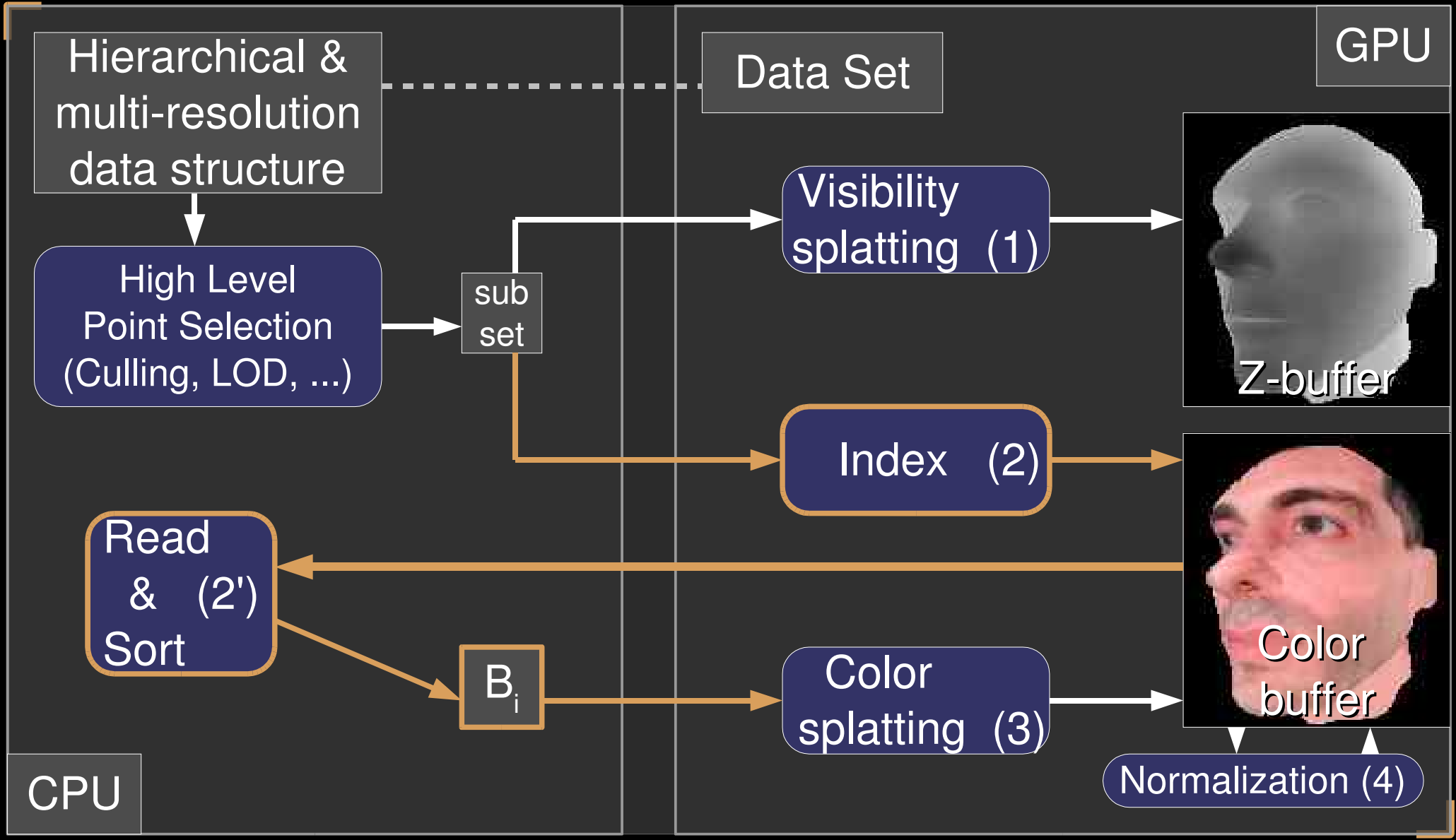
Accurate Point Selection



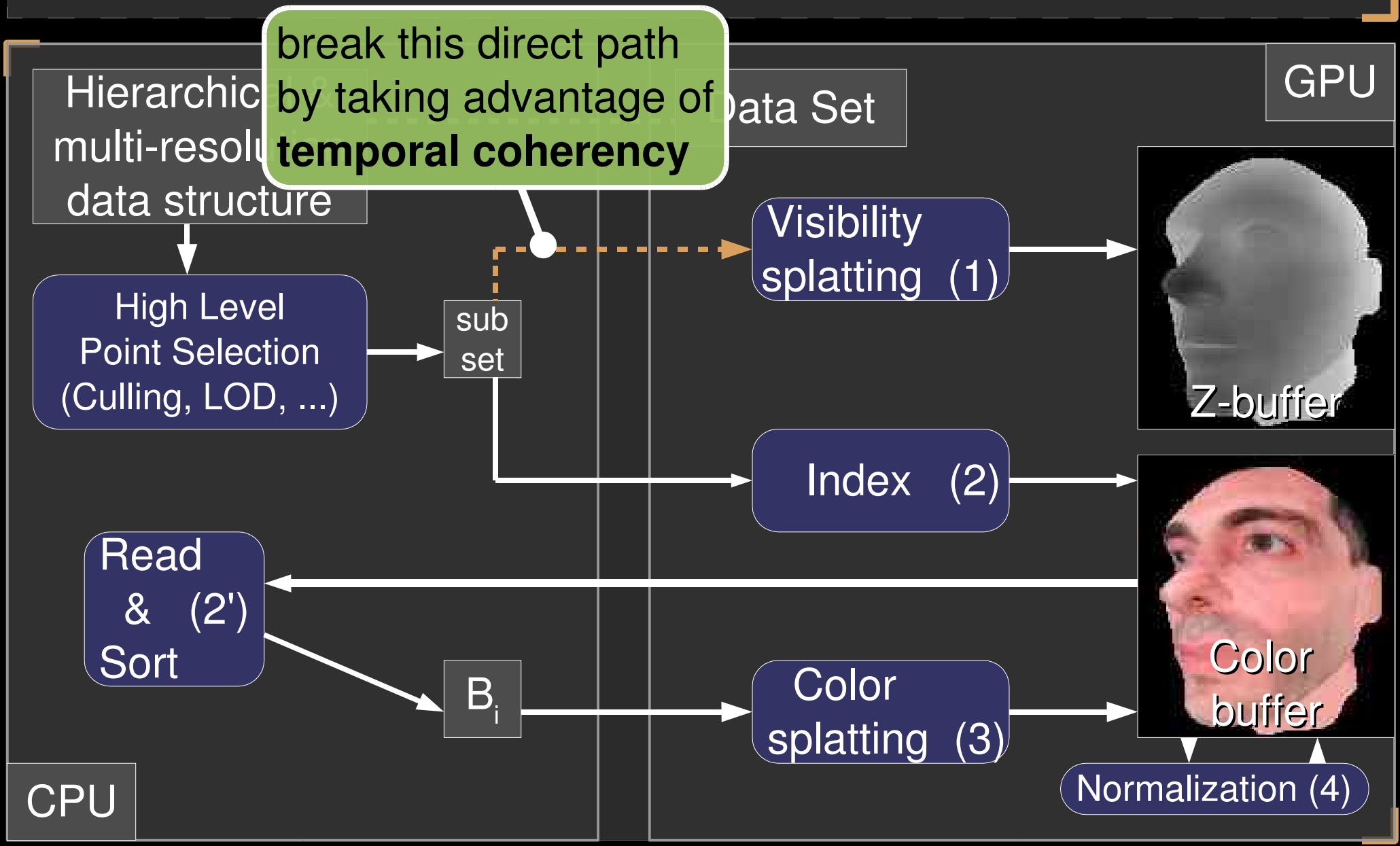
Accurate Point Selection



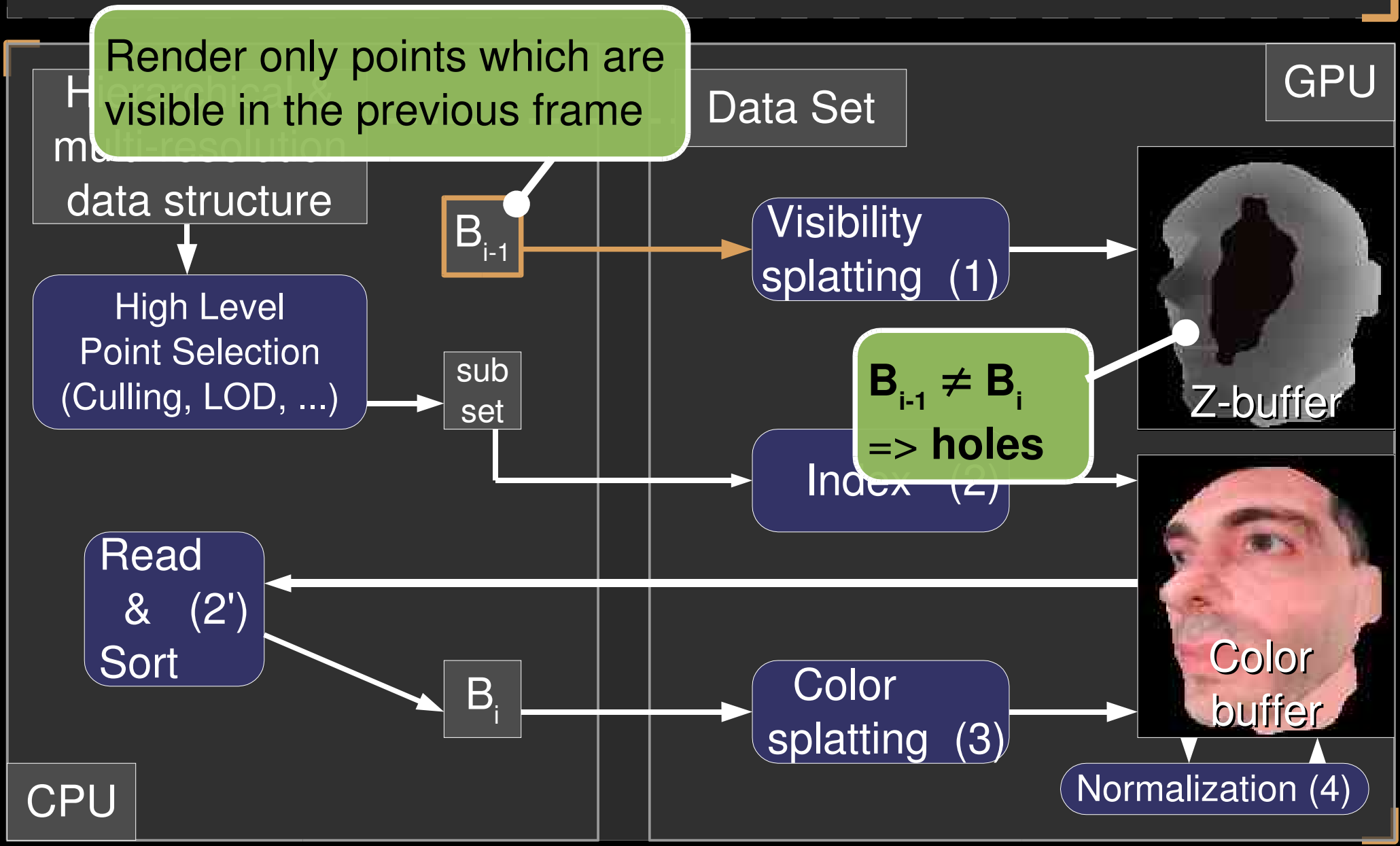
Accurate Point Selection



Accurate Point Selection

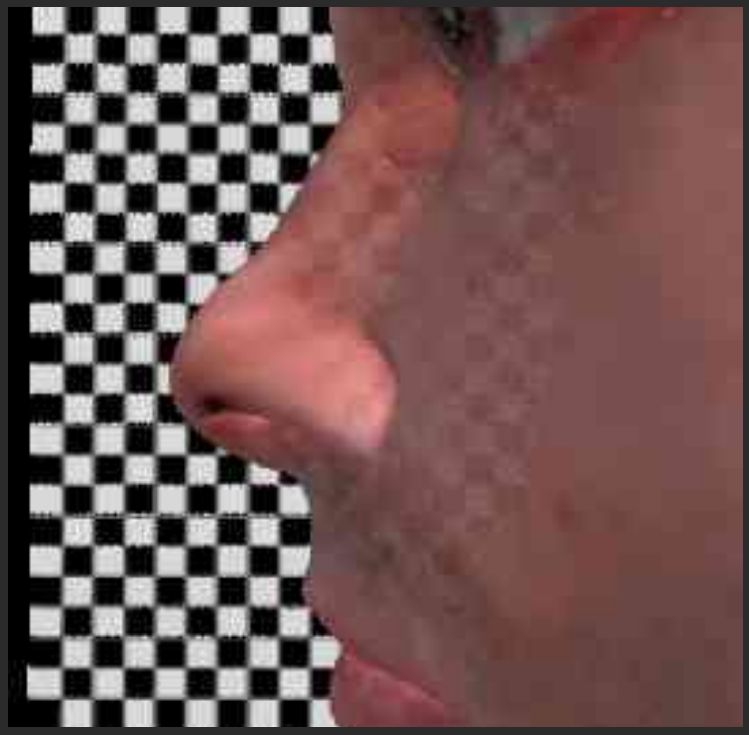


Accurate Point Selection



Temporal Coherency : Artifacts

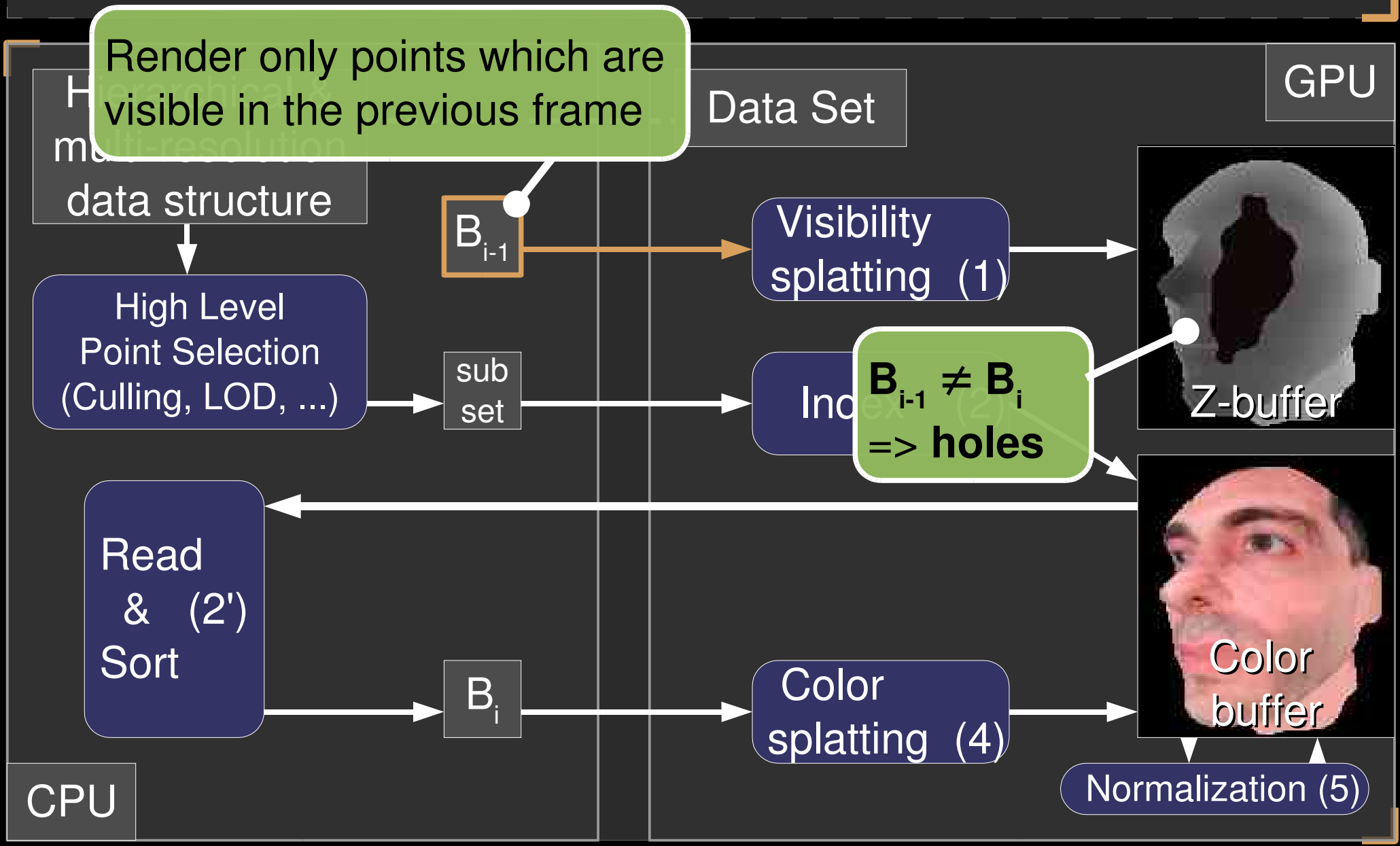
Frame i



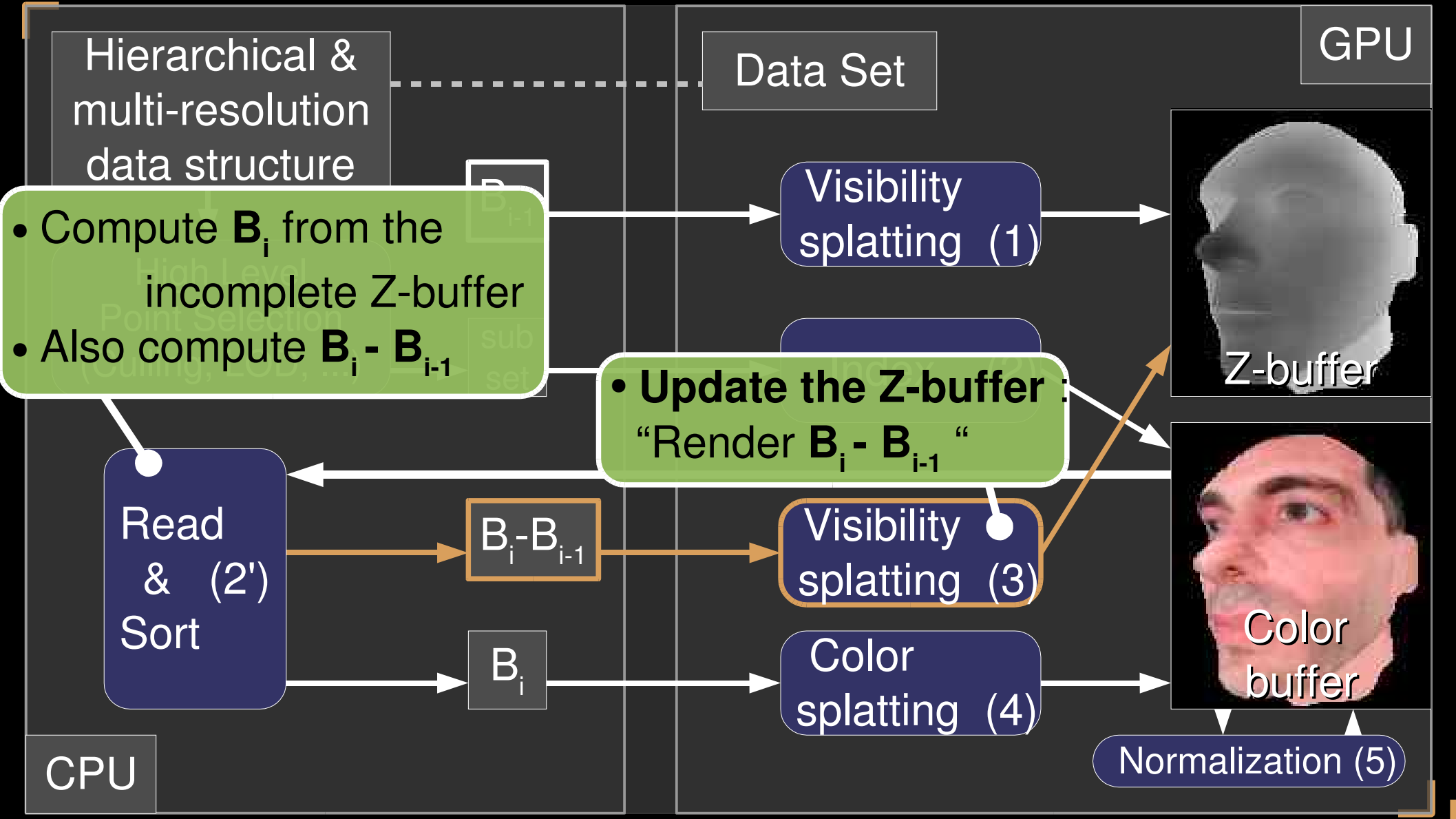
Frame i+1

temporal coherency approximation leads to artifacts

Temporal Coherency

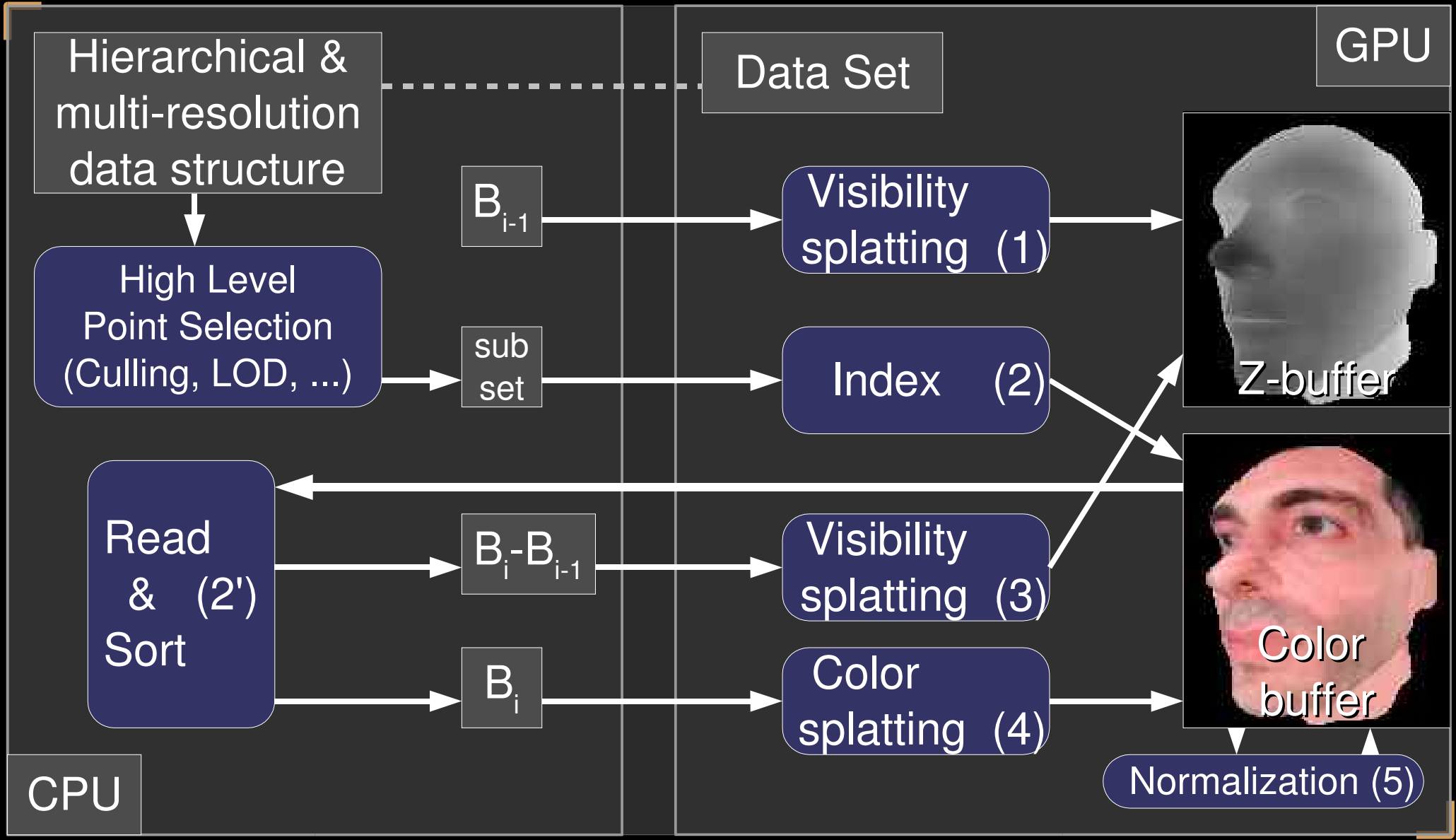


Temporal Coherency



The Complete Algorithm

summary step by step



One point per pixel ...

- Deferred Splatting allows only one point per pixel
- Advantages
  - Remove superfluous points (LOD selection)
  - Solve color buffer overflow (only 8 bits per component)
- *Drawbacks*

One point per pixel ...

- Deferred Splatting allows only one point per pixel

- *Advantages*
- Drawbacks
  - We may lose texture information
  - High frequency textured models + coarse high-level LOD selection

    ➔ flickering artifacts ...
  - Can be solved using *surfel mipmap* [Pfister et al. 00]

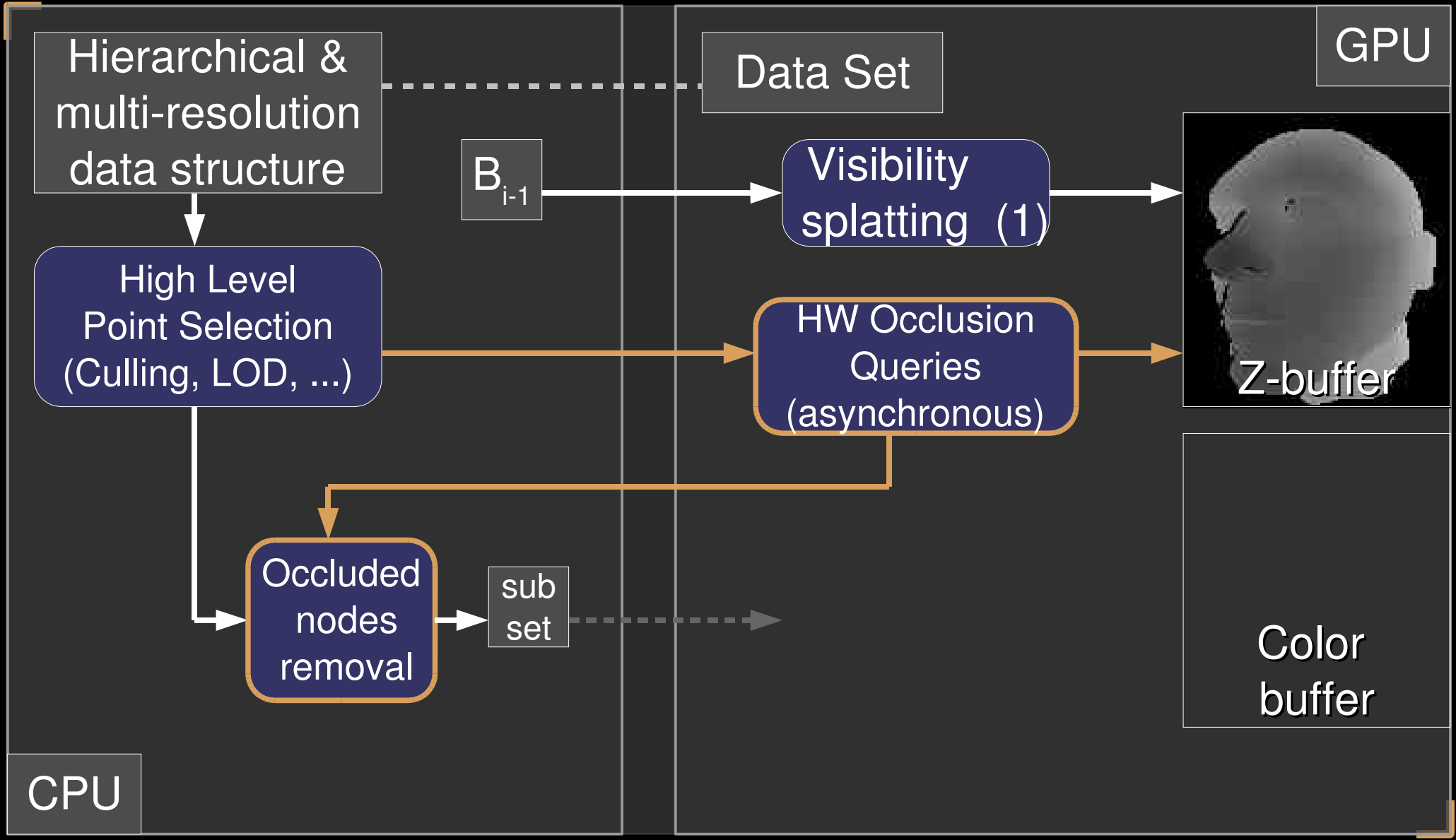
Deferred Splatting

Applications

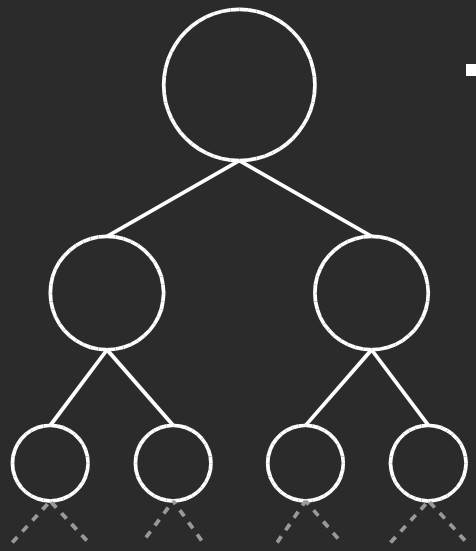
Occlusion Culling

Sequential Point Trees

High-Level **Occlusion Culling**



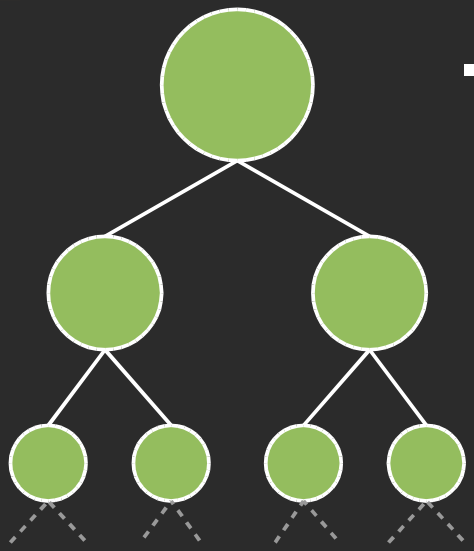
Sequential Point Trees [Dachsbacher03]



Preprocessing:

build a sequential version of the hierarchy

Sequential Point Trees [Dachsbacher03]



Preprocessing:

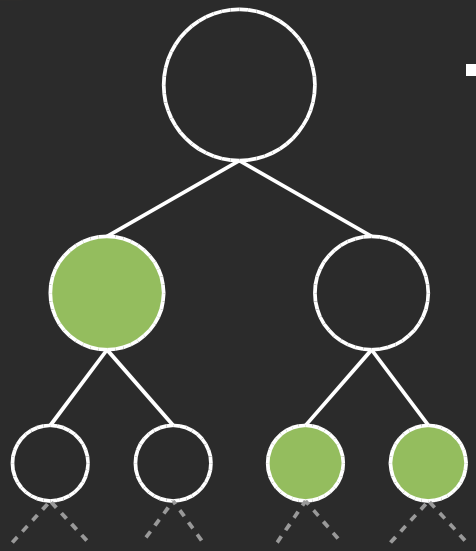
build a sequential version of the hierarchy

Rendering:

CPU: fast & coarse selection of a prefix

GPU: fine LOD selection at the point level

Sequential Point Trees [Dachsbacher03]



Preprocessing:

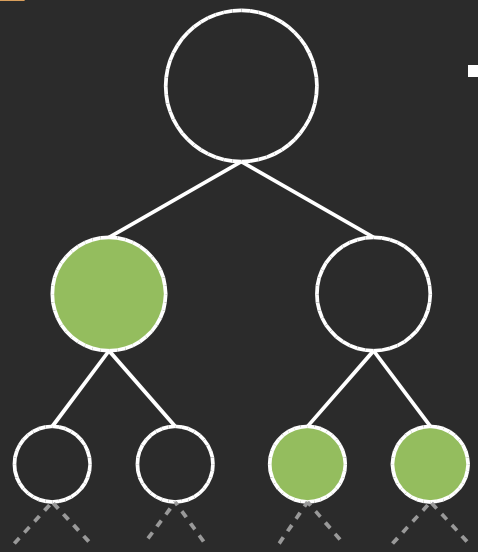
build a sequential version of the hierarchy

Rendering:

CPU: fast & coarse selection of a prefix

GPU: fine LOD selection at the point level

Sequential Point Trees



Preprocessing:

build a sequential version of the hierarchy

Rendering:

CPU: fast & coarse selection of a prefix

GPU: fine LOD selection at the point level

SPT

Coarse SPT selection

prefix

CPU

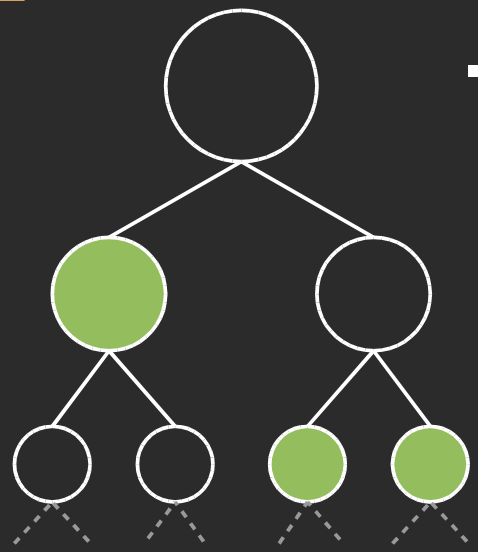
GPU

Visibility splatting(1)
+
SPT fine selection

Color splatting (2)
+
SPT fine selection

Classical High-Quality Splatting:
all points of the coarse prefix are processed by **2 complex** vertex programs
=> inefficient

Sequential Point Trees



Preprocessing:

build a sequential version of the hierarchy

Rendering:

CPU: fast & coarse selection of a prefix

GPU: fine LOD selection at the point level

SPT

Coarse SPT selection

prefix

GPU

Index (2)

+

SPT fine selection

CPU

Deferred Splatting:

all points of the coarse prefix are processed by **1 very simple** vertex program

=> efficient

Results

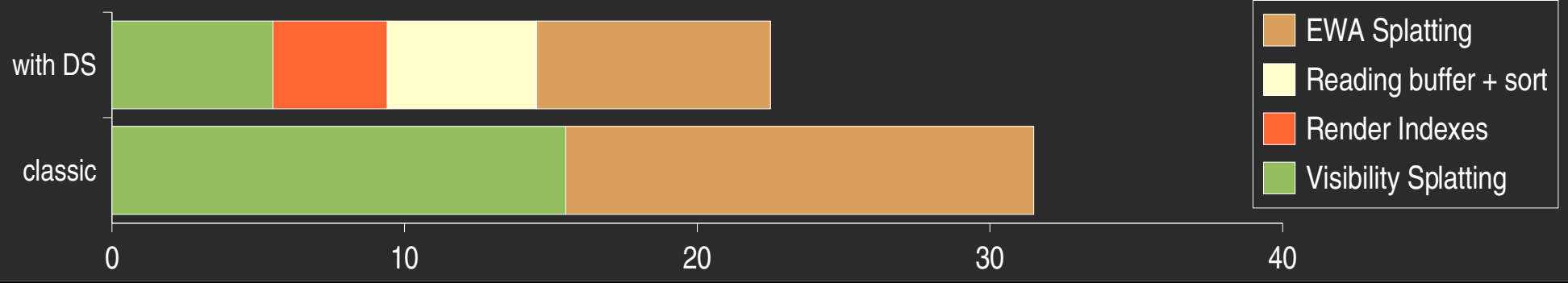
Classical GPU based High-Quality Splatting

versus

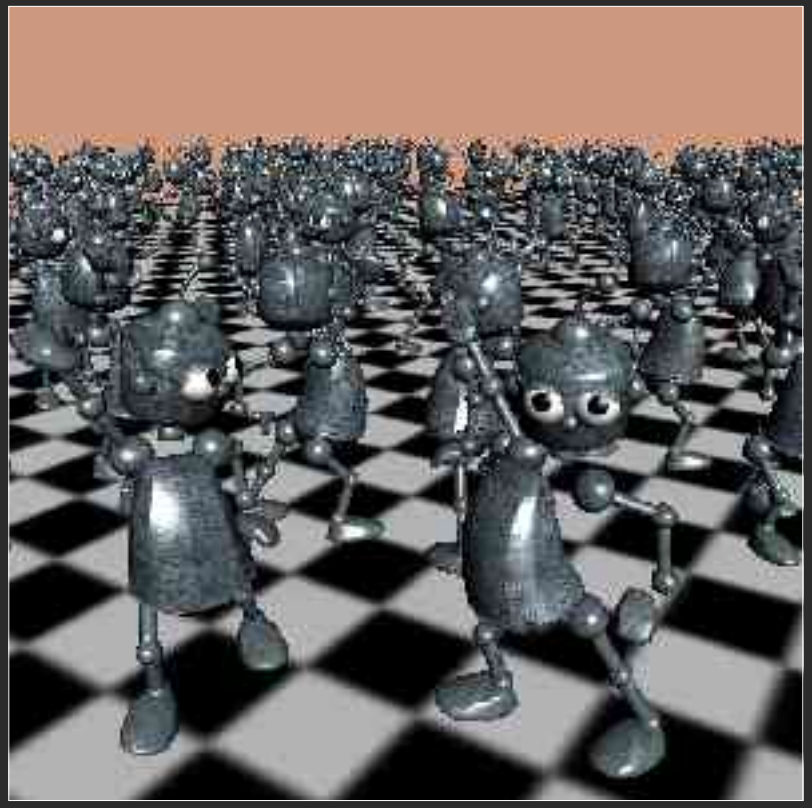
Deferred Splatting

Results : Simple Head

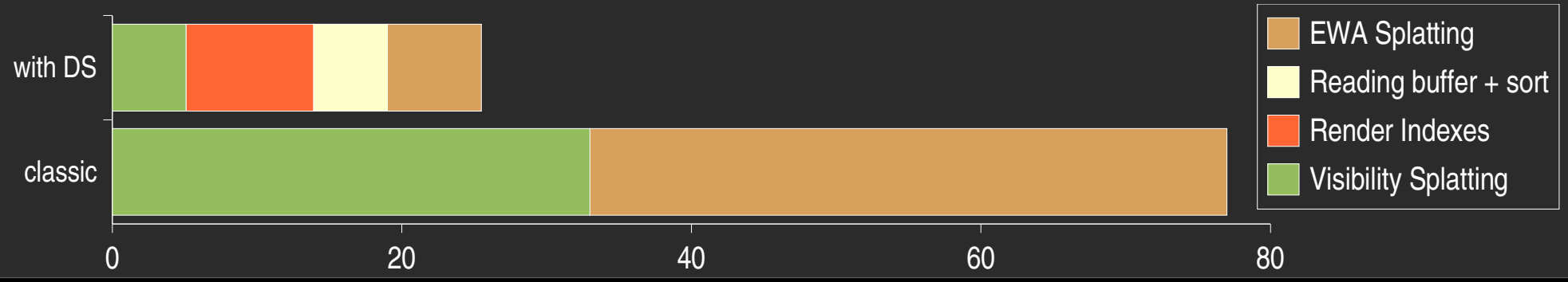
- 285k points
- Average FPS:
  - EWA Splatting: 34
  - Deferred Splatting: 41
- Speed up: x1.2
- % of culled points: 50-70%



Results : 200 Hugo

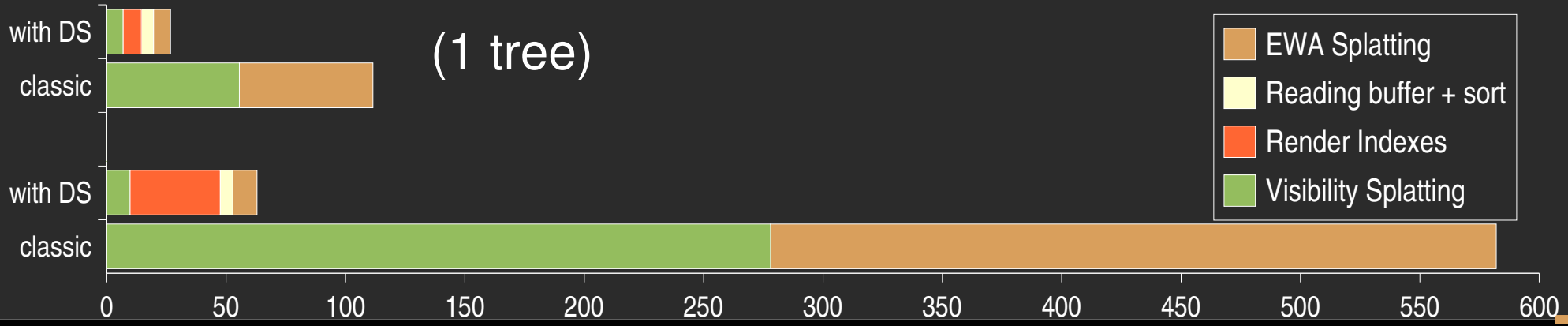


- 1 Hugo = 450k points
- Scene = 200 Hugo in motion
- Average FPS:
  - EWA Splatting: 11.5
  - Deferred Splatting: 34.5
- Speed up : x3
- % of culled points: 90%



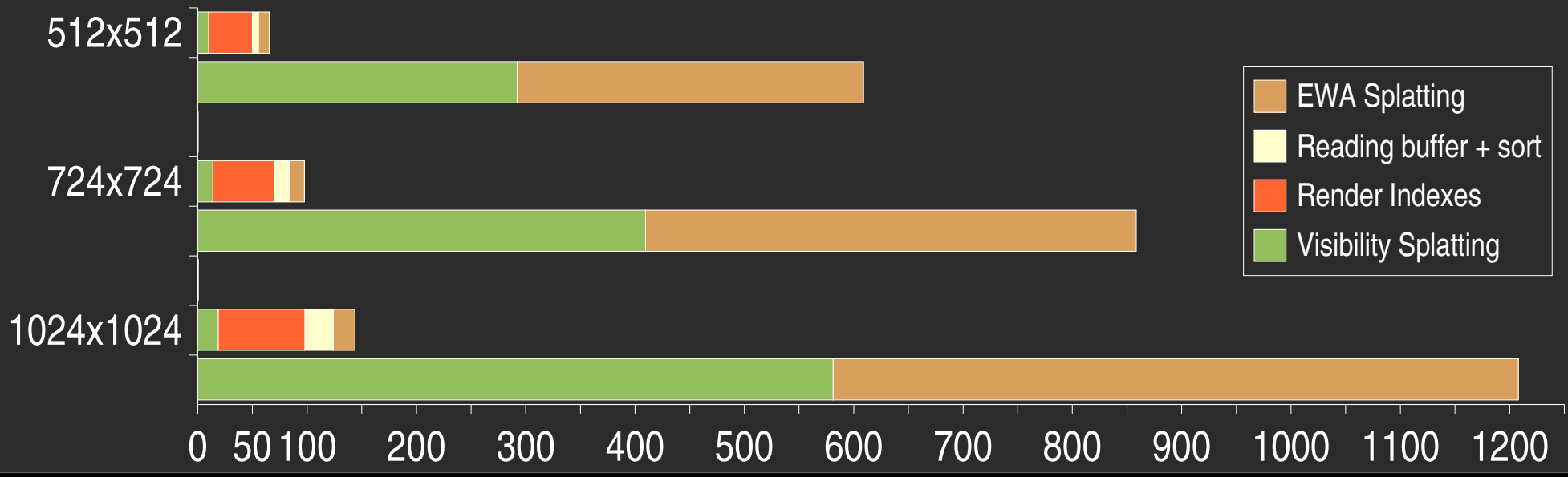
Results : Forest

- 1 tree = 750k points
- Scene = 6800 trees
- Average FPS:
  - EWA Splatting: 1.1-1.8
  - Deferred Splatting:11-20
- Speed up : x10
- % of culled points: 90-97%



What about screen resolutions ?

- When the screen size increases
  - The rendering time linearly increases
  - The speed up of deferred splatting remains constant
- Large resolution => reading the color buffer becomes expensive: 1024^2 => 25ms !
  - AGP limitation -> PCI express ?



Usability

- Unsuitable for simple scenes (< ~300k points)
- Based on the assumption that a point is visible or not
  - true for small points only (< ~10 pixels)
  - For our initial context it is always true
    - large points are inefficient => use triangles
  - If you don't have a polygonal representation:
    - render large points anyway

Conclusion

- works at the point level and does:
  - view frustum culling
  - occlusion culling (and back-face culling)
  - LOD selection
- high quality splatting on highly complex scenes
- suitable for dynamic scenes & point clouds
- no assumption on the high-level data structure
- no additional preprocessing
- simple and efficient

Future Works

- Full hardware implementation
  - keep the CPU free
  - no slow reading from the GPU to the CPU
- More efficient/accurate high-level point selection
  - new data structures
  - new algorithms

Questions ?