CHENHAO XIE*, SHUAIWEN LEON SONG**, JING WANG***, WEIGONG ZHANG***, XIN FU*

* ECE DEPARTMENT, UNIVERSITY OF HOUSTON
** HPC GROUP, PACIFIC NORTHWEST NATIONAL LAB (PNNL)
*** CAPITAL NORMAL UNIVERSITY BEIJING

PRESENTED AT: HPCA 2017

REVIEW: ANDREW DOBIS

# PROCESSING-IN-MEMORY ENABLED GRAPHICS PROCESSORS FOR 3D RENDERING

# EXECUTIVE SUMMARY

- **Motivation**: Texture filtering = **60%** of **memory requests** in real-time rendering applications.

# EXECUTIVE SUMMARY

- **Motivation**: Texture filtering = **60%** of **memory requests** in real-time rendering applications.
- **Problem**: Real-time rendering is a highly memory bound task due to texture filtering.

# EXECUTIVE SUMMARY

- **Motivation**: Texture filtering = **60%** of **memory requests** in real-time rendering applications.
- **Problem**: Real-time rendering is a highly memory bound task due to texture filtering.
- **Idea**: Bypass memory bottleneck caused by texture filtering by:
  - Using **more** *memory efficient* filtering pipeline.
  - **Reducing** data movement in system.

# EXECUTIVE SUMMARY

- **Motivation**: Texture filtering = **60%** of **memory requests** in real-time rendering applications.
- **Problem**: Real-time rendering is a highly memory bound task due to texture filtering.
- **Idea**: Bypass memory bottleneck caused by texture filtering by:
  - Using **more** *memory efficient* filtering pipeline.
  - **Reducing** data movement in system.
- **Solution**: Advanced-Texture Filtering In Memory (A-TFIM).
  - Implements a portion of the **texture filtering pipeline** in **HMC**.
  - Uses a **camera-angle threshold** for **performance/accuracy** ratio control.

# EXECUTIVE SUMMARY

- **Motivation**: Texture filtering = **60%** of **memory requests** in real-time rendering applications.
- **Problem**: Real-time rendering is a highly memory bound task due to texture filtering.
- **Idea**: Bypass memory bottleneck caused by texture filtering by:
  - Using **more** *memory efficient* filtering pipeline.
  - **Reducing** data movement in system.
- **Solution**: Advanced-Texture Filtering In Memory (A-TFIM).
  - Implements a portion of the **texture filtering pipeline** in **HMC**.
  - Uses a **camera-angle threshold** for **performance/accuracy** ratio control.
- **Key results**: Evaluated on various 3D video-games.
  - Average **1.4x** rendering speedup over baseline.
  - Average **22% less** energy consumption compared to baseline.

# OUTLINE

- Background: Texture Filtering
- Motivation
- Solutions:
  - B-PIM: Basic Processing In Memory
  - S-TFIM: Simple Texture Filtering In Memory
  - A-TFIM: Advanced Texture Filtering In Memory
- Evaluation Methodology
- Evaluation Results
- Conclusion

# OUTLINE

- Background: Texture Filtering
- Motivation
- Solutions:
  - B-PIM: Basic Processing In Memory
  - S-TFIM: Simple Texture Filtering In Memory
  - A-TFIM: Advanced Texture Filtering In Memory
- Evaluation Methodology
- Evaluation Results
- Conclusion

# BACKGROUND: 3D RENDERING

- Goal: Convert a representation of a scene into an image.

# BACKGROUND: 3D RENDERING

- Goal: Convert a representation of a scene into an image.
- Used for 3D video-games and movies.

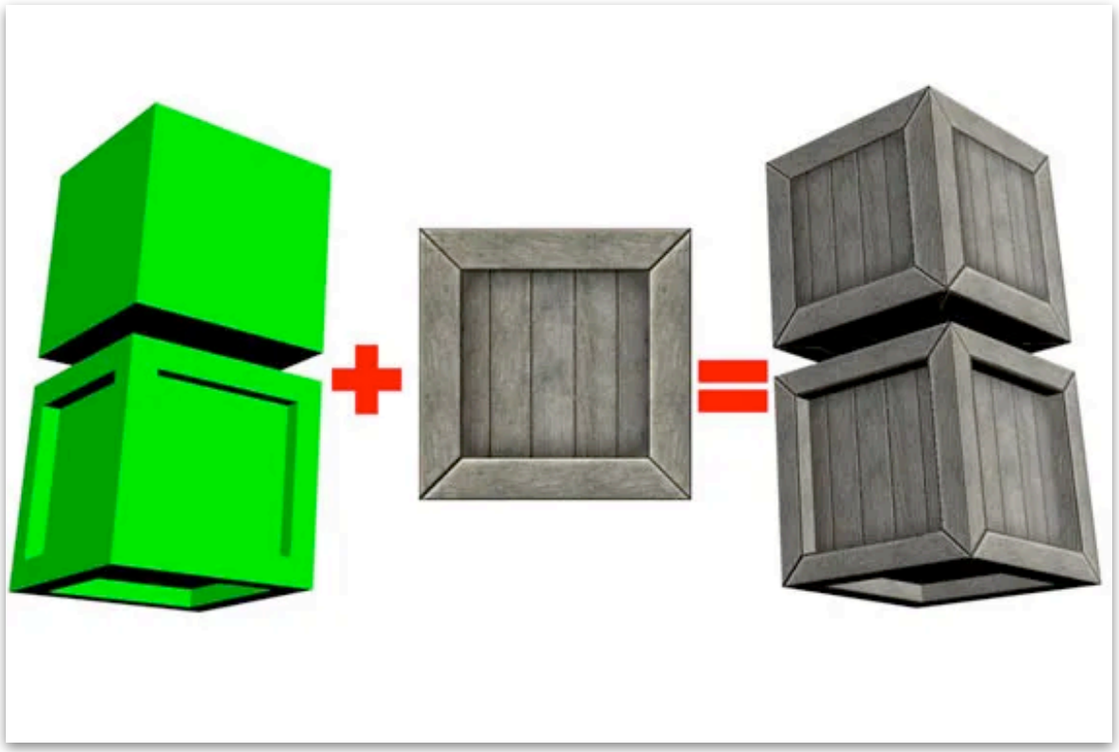
# BACKGROUND: 3D RENDERING

- Goal: Convert a representation of a scene into an image.
- Used for 3D video-games and movies.
- Textures: Used to add pre-computed color details.

# BACKGROUND: 3D RENDERING

- Goal: Convert a representation of a scene into an image.
- Used for 3D video-games and movies.
- Textures: Used to add pre-computed color details.

**Texturing Example**



*[1]: https://www.gamedeveloper.com/design/book-excerpt-3d-game-textures-create-professional-game-art-using-photoshop*

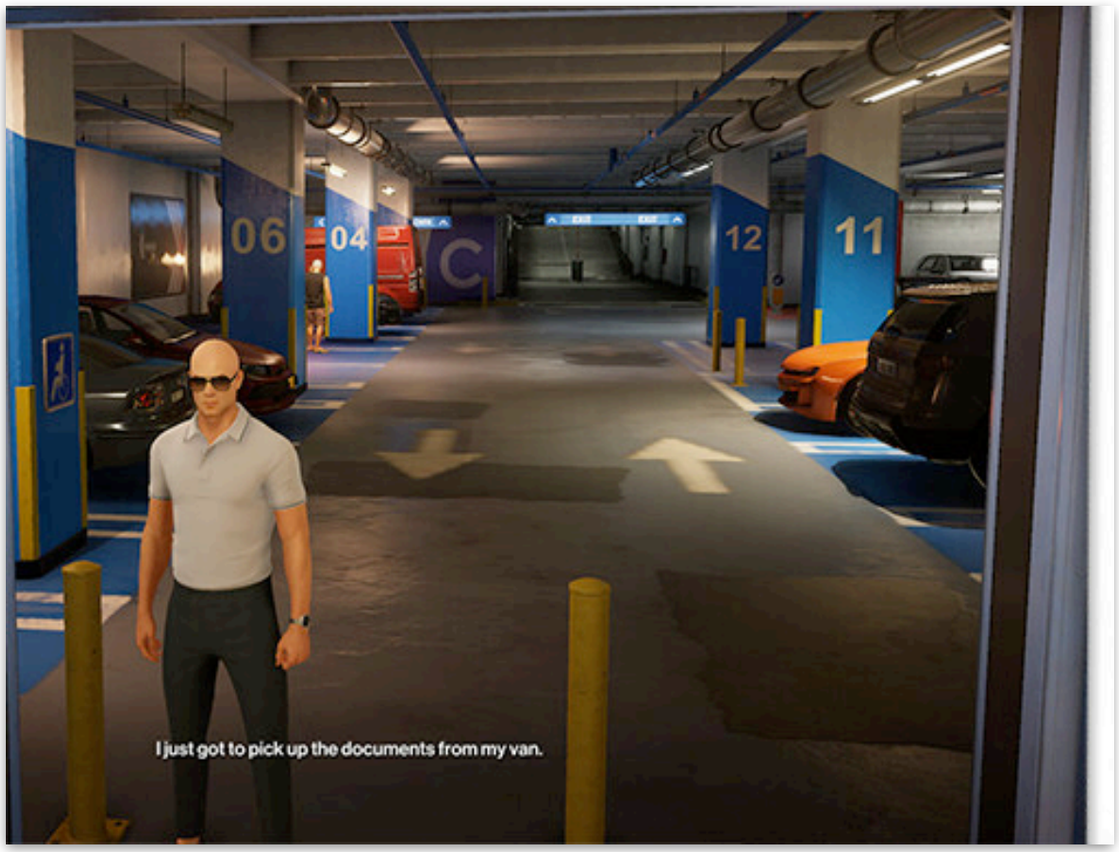
# BACKGROUND: TEXTURE FILTERING

- Problem: Greatly reduced image sharpness if sample resolution >> texture resolution.

# BACKGROUND: TEXTURE FILTERING

- Problem: Greatly reduced image sharpness if sample resolution >> texture resolution.

**No Filtering**

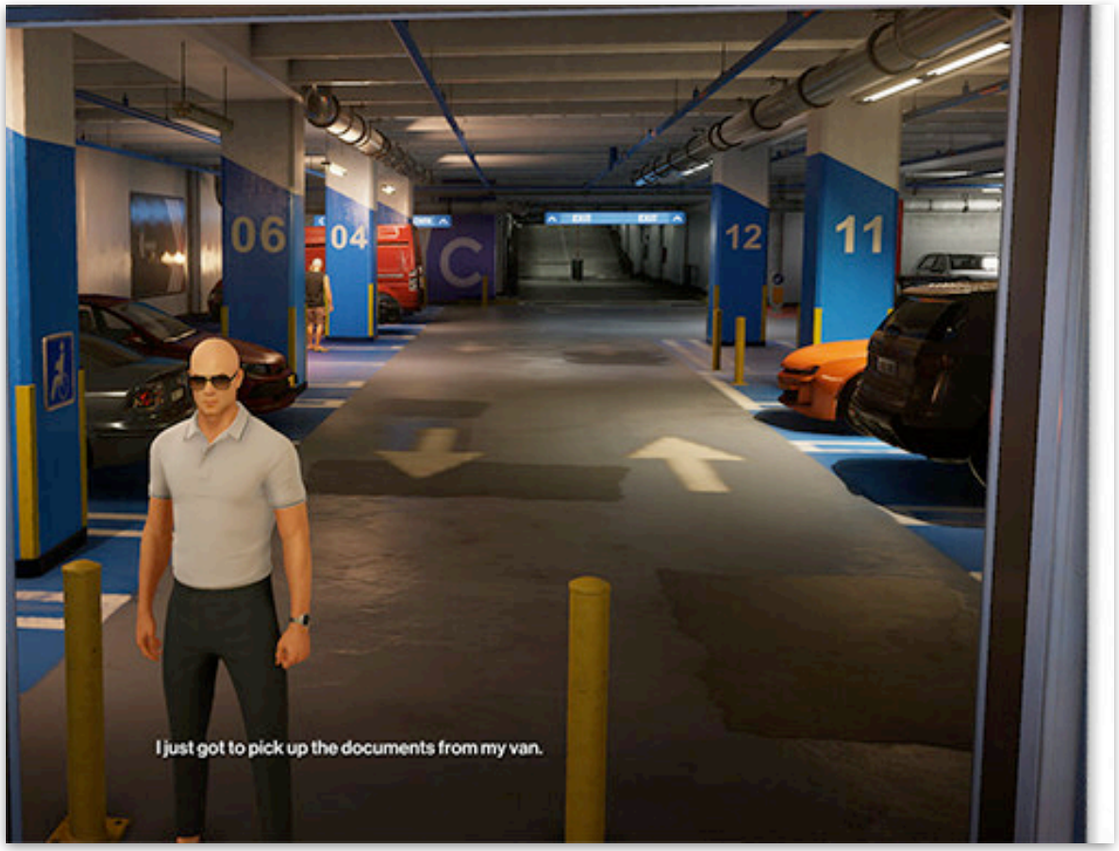



*[7]: https://www.game-debate.com/news/27720/what-do-texture-filtering-and-anisotropic-filtering-do-graphics-settings-explained*

# BACKGROUND: TEXTURE FILTERING

- Problem: Greatly reduced image sharpness if sample resolution >> texture resolution.
- **Solution**: Texture Filtering.

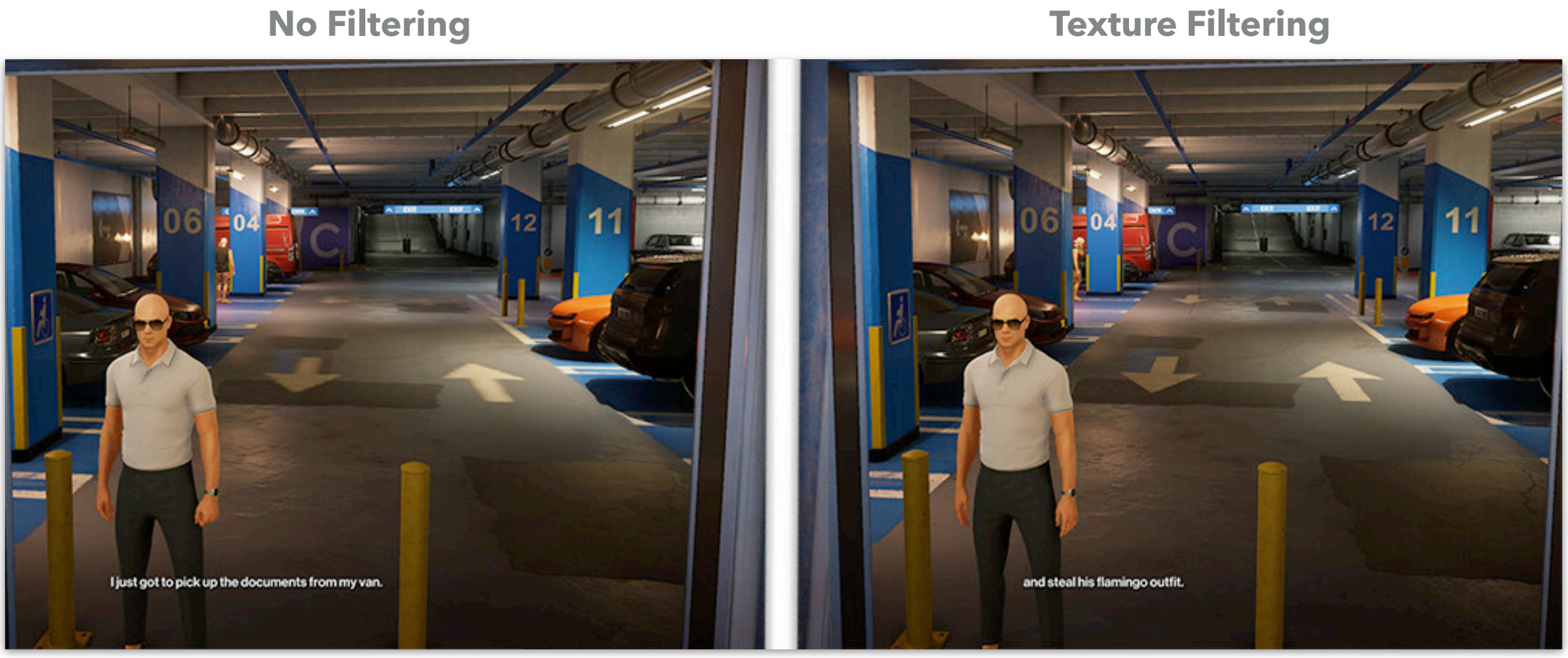
No Filtering




*[7]: https://www.game-debate.com/news/27720/what-do-texture-filtering-and-anisotropic-filtering-do-graphics-settings-explained*

# BACKGROUND: TEXTURE FILTERING

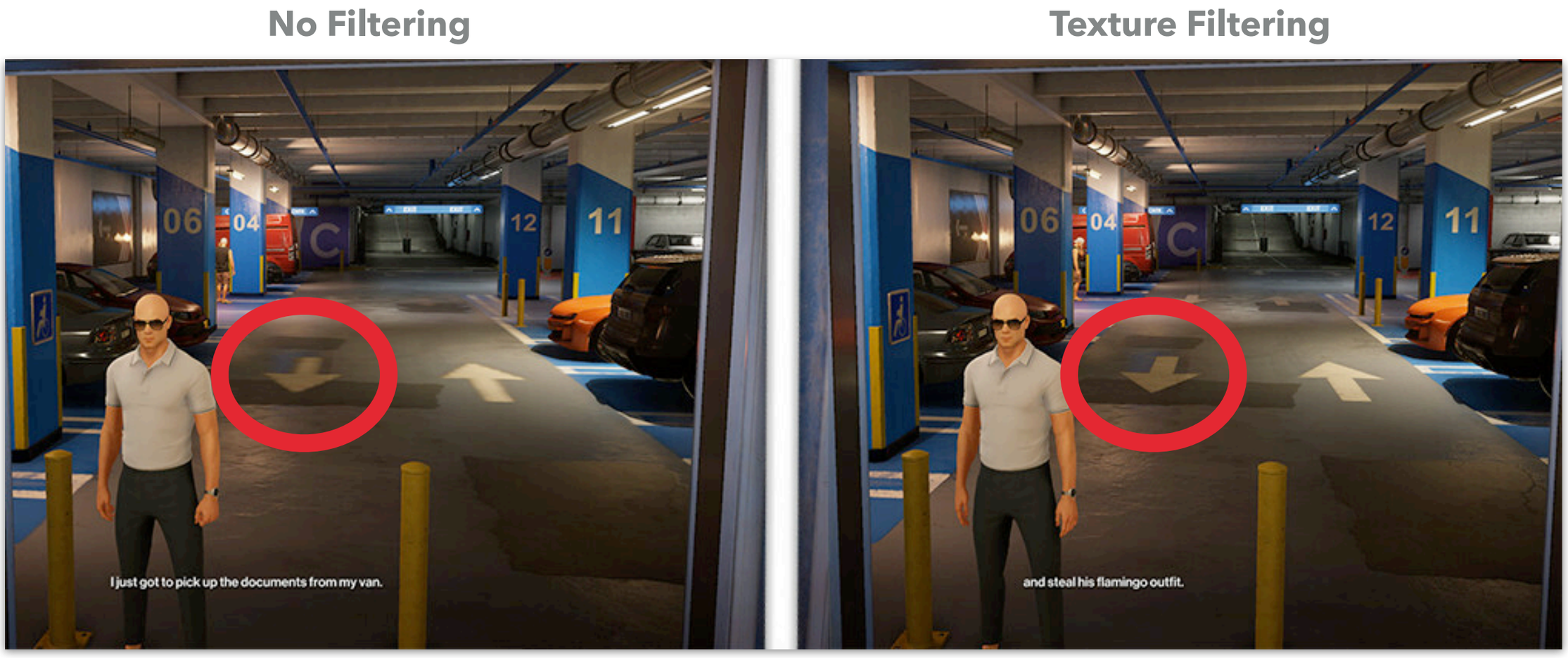
- Problem: Greatly reduced image sharpness if sample resolution >> texture resolution.
- **Solution**: Texture Filtering.




*[7]: https://www.game-debate.com/news/27720/what-do-texture-filtering-and-anisotropic-filtering-do-graphics-settings-explained*

# BACKGROUND: TEXTURE FILTERING

- Problem: Greatly reduced image sharpness if sample resolution >> texture resolution.
- **Solution**: Texture Filtering.




*[7]: https://www.game-debate.com/news/27720/what-do-texture-filtering-and-anisotropic-filtering-do-graphics-settings-explained*

# BACKGROUND: TEXTURE FILTERING

- Problem: Greatly reduced image sharpness if sample resolution >> texture resolution.
- **Solution**: Texture Filtering.

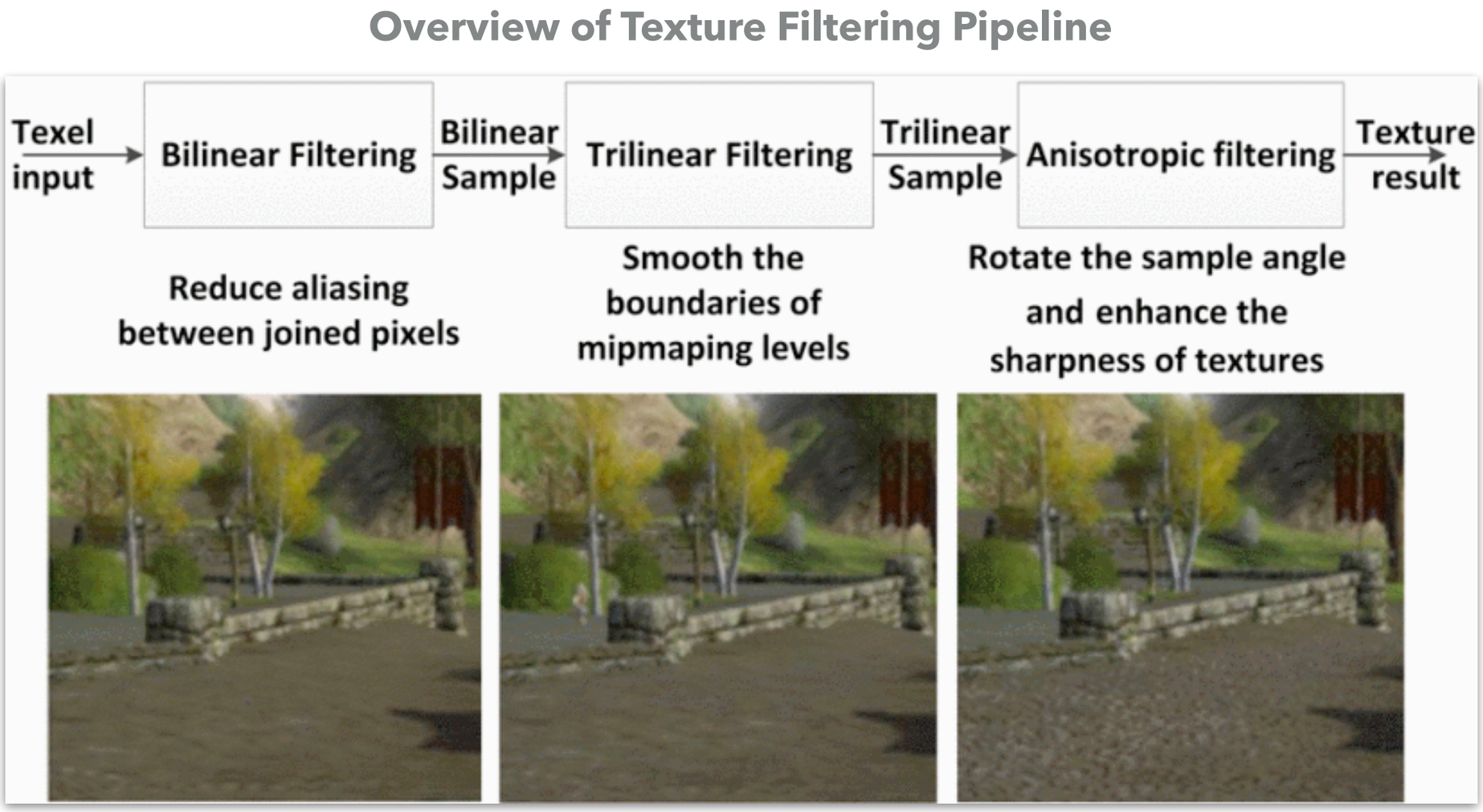
**No Filtering**

**Texture Filtering**

**Image is blurry when camera angle is low.**

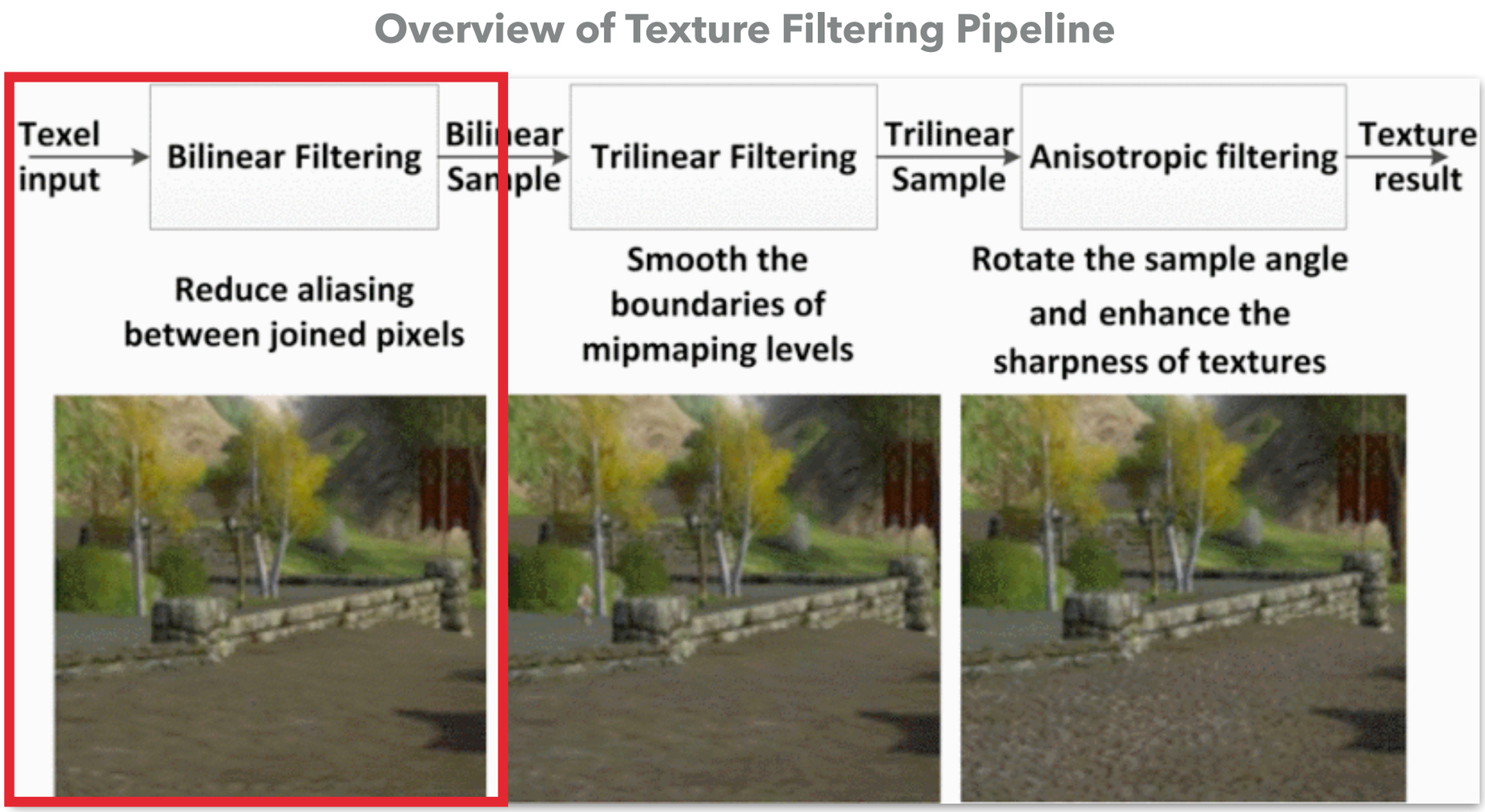
# BACKGROUND: TEXTURE FILTERING

- Goal: Reduce blur from under-sampling textures



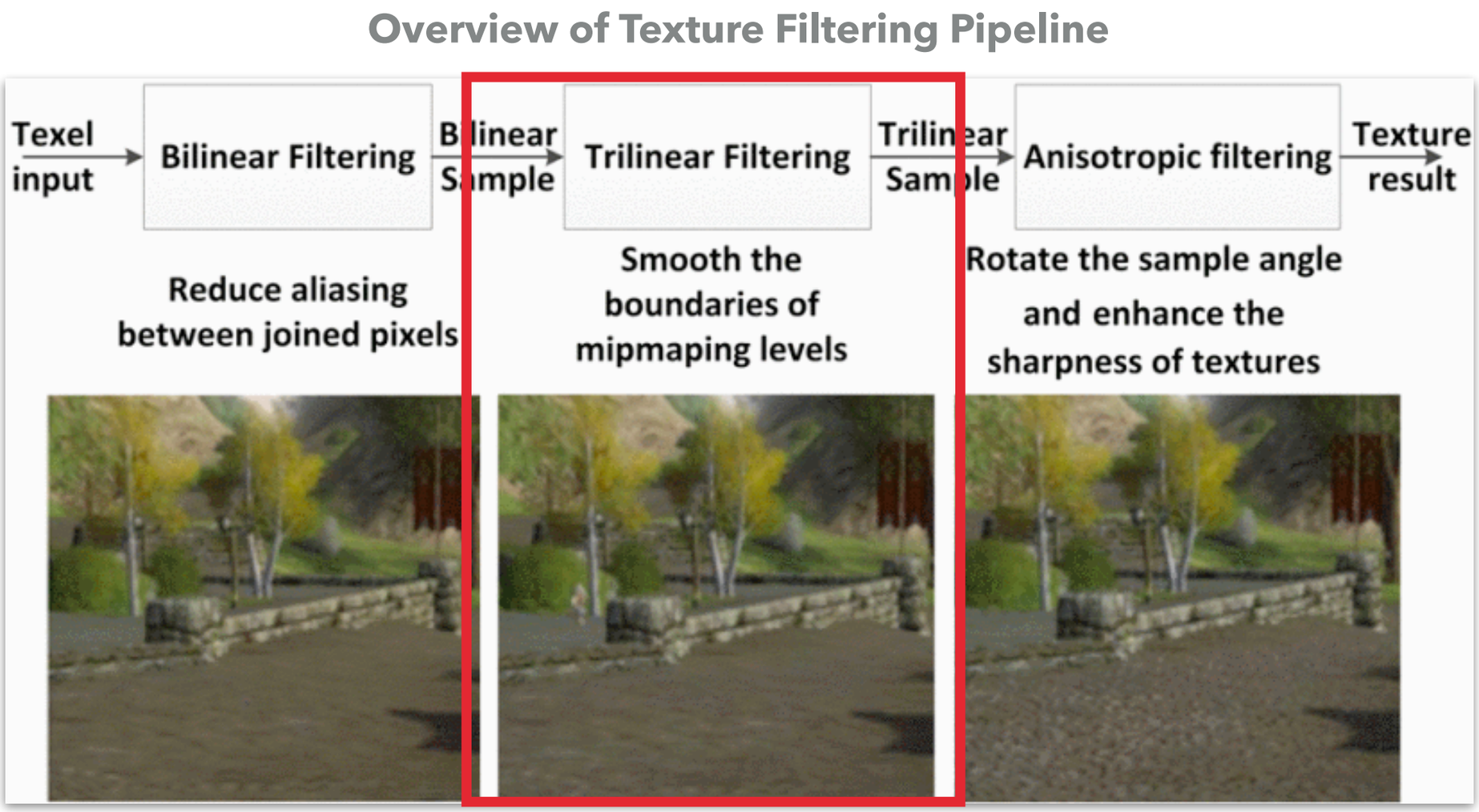
# BACKGROUND: TEXTURE FILTERING

- Goal: Reduce blur from under-sampling textures



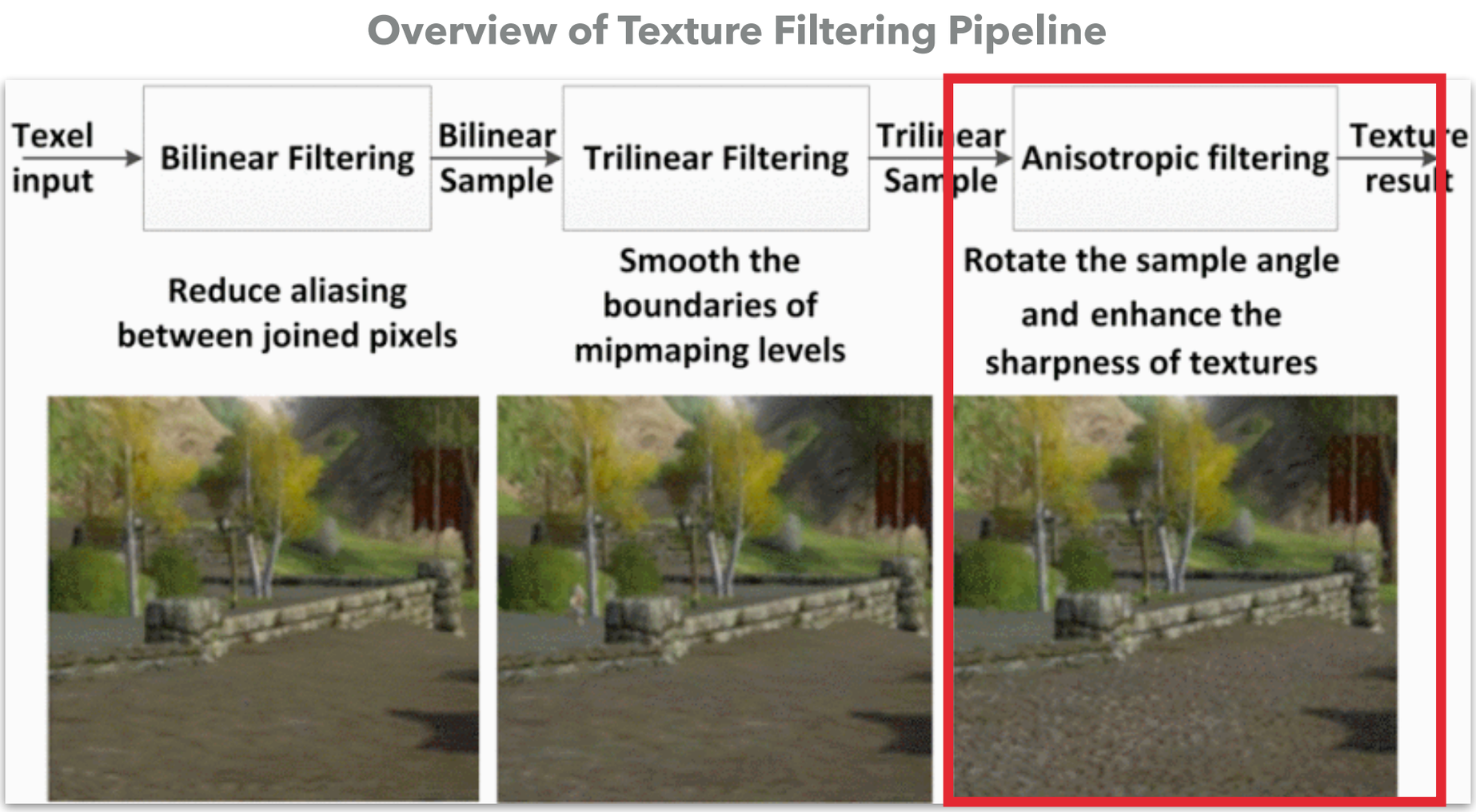
# BACKGROUND: TEXTURE FILTERING

- Goal: Reduce blur from under-sampling textures



# BACKGROUND: TEXTURE FILTERING

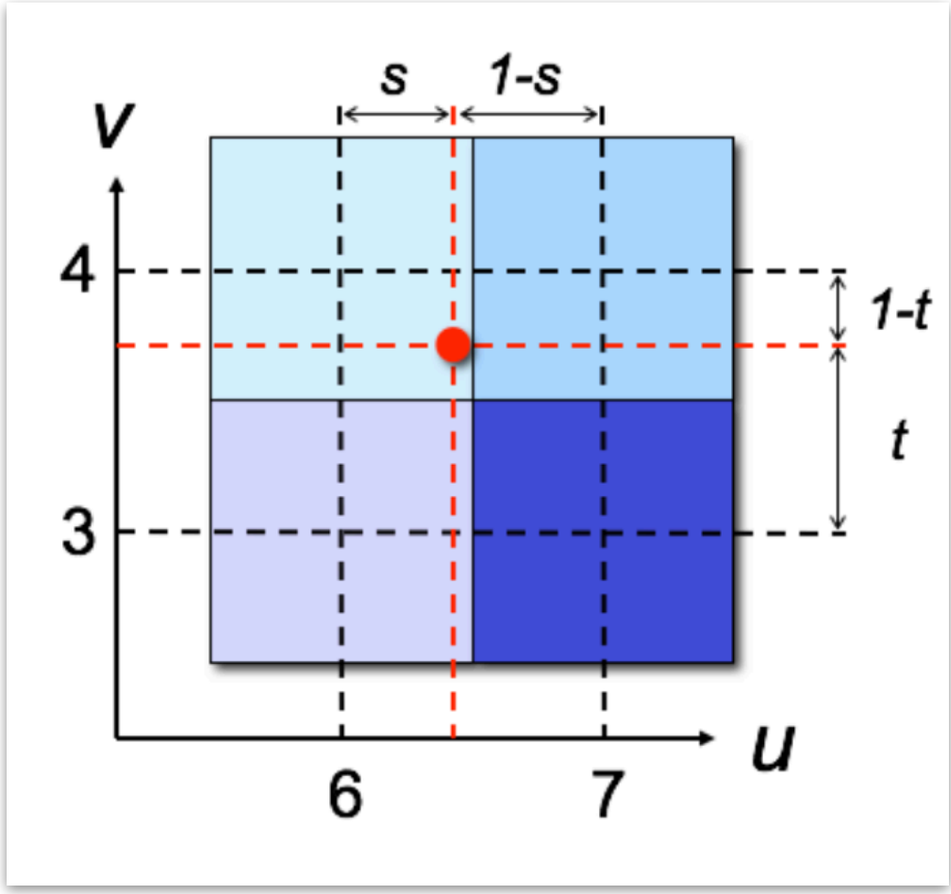
- Goal: Reduce blur from under-sampling textures



# BACKGROUND: BILINEAR FILTERING

- Texture coordinates: (u,v) pair defines a texel coordinate.

**Example 2x2 texture being sampled**

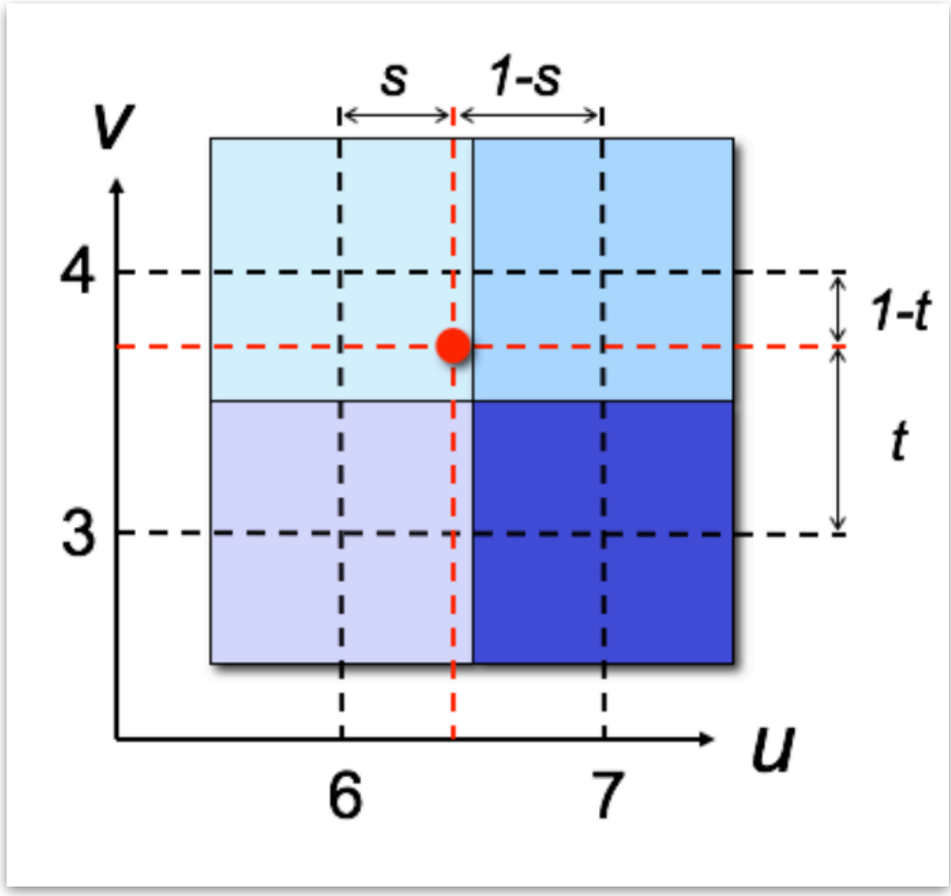



*[3]: Introduction to Computer Graphics (Spring 2020), EPFL, Prof. Mark Pauly, Lecture 7a: Texturing, Slide 20*

# BACKGROUND: BILINEAR FILTERING

- Texture coordinates: (u,v) pair defines a texel coordinate.
- Bilinear filtering: Sample texel and neighboring texels.

**Example 2x2 texture being sampled**

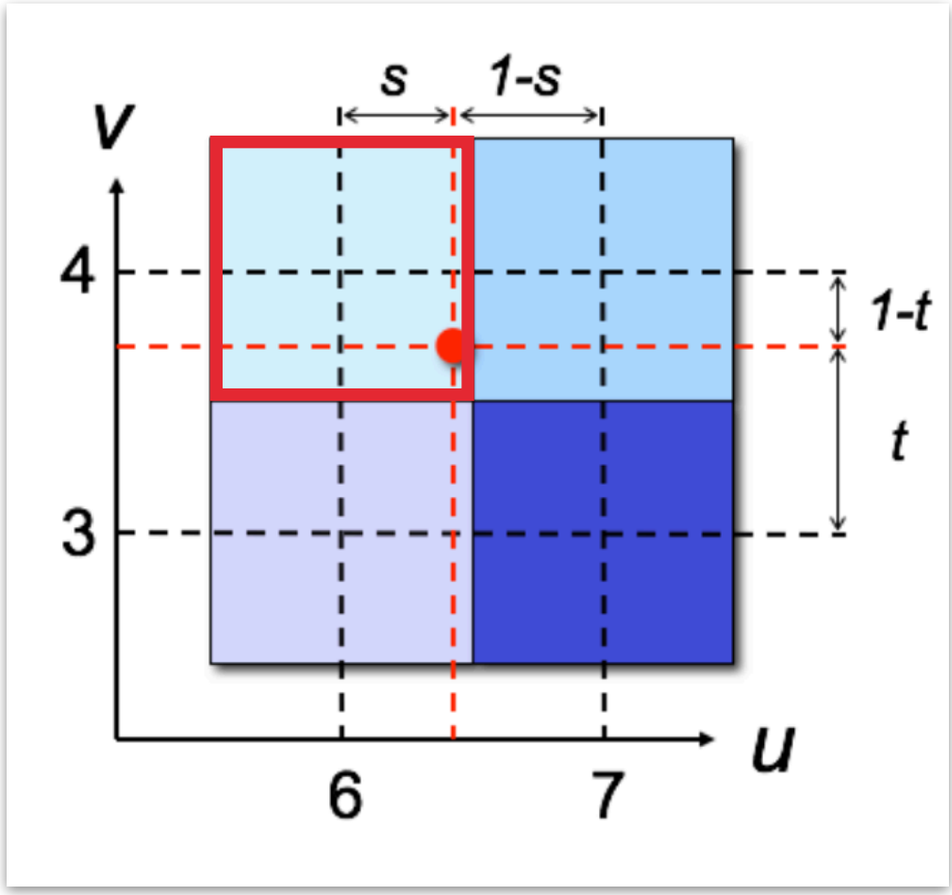



*[3]: Introduction to Computer Graphics (Spring 2020), EPFL, Prof. Mark Pauly, Lecture 7a: Texturing, Slide 20*

# BACKGROUND: BILINEAR FILTERING

- Texture coordinates: (u,v) pair defines a texel coordinate.
- Bilinear filtering: Sample texel and neighboring texels.

**Example 2x2 texture being sampled**

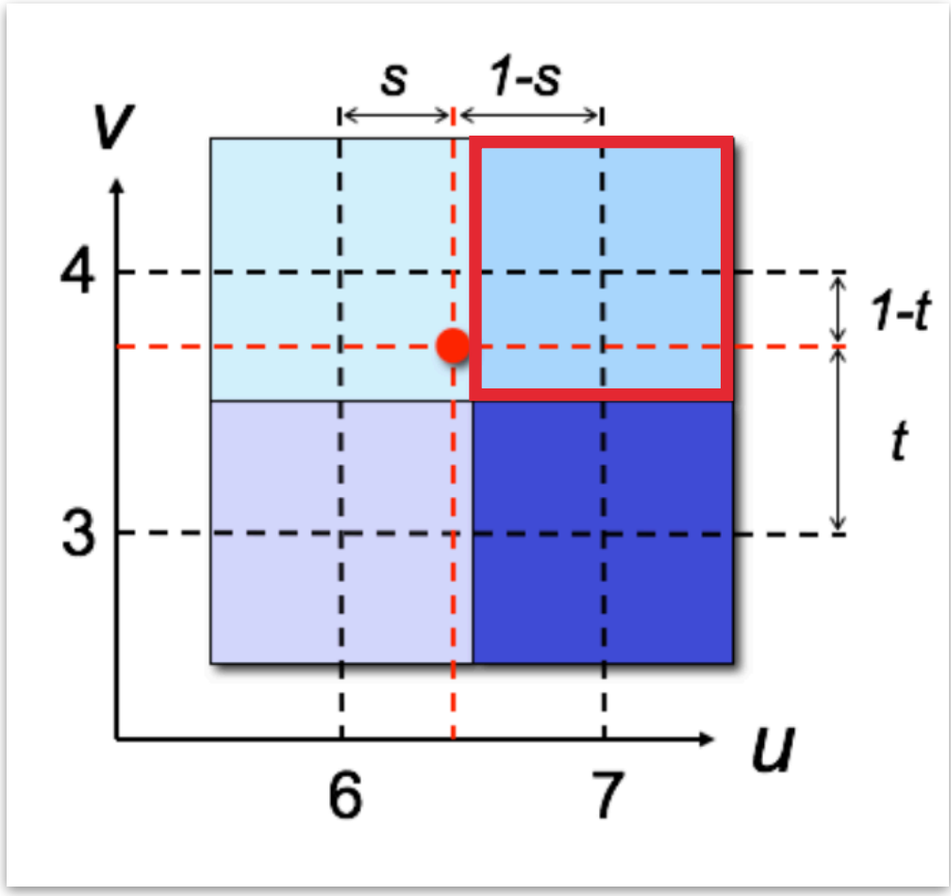



*[3]: Introduction to Computer Graphics (Spring 2020), EPFL, Prof. Mark Pauly, Lecture 7a: Texturing, Slide 20*

# BACKGROUND: BILINEAR FILTERING

- Texture coordinates: (u,v) pair defines a texel coordinate.
- Bilinear filtering: Sample texel and neighboring texels.

**Example 2x2 texture being sampled**

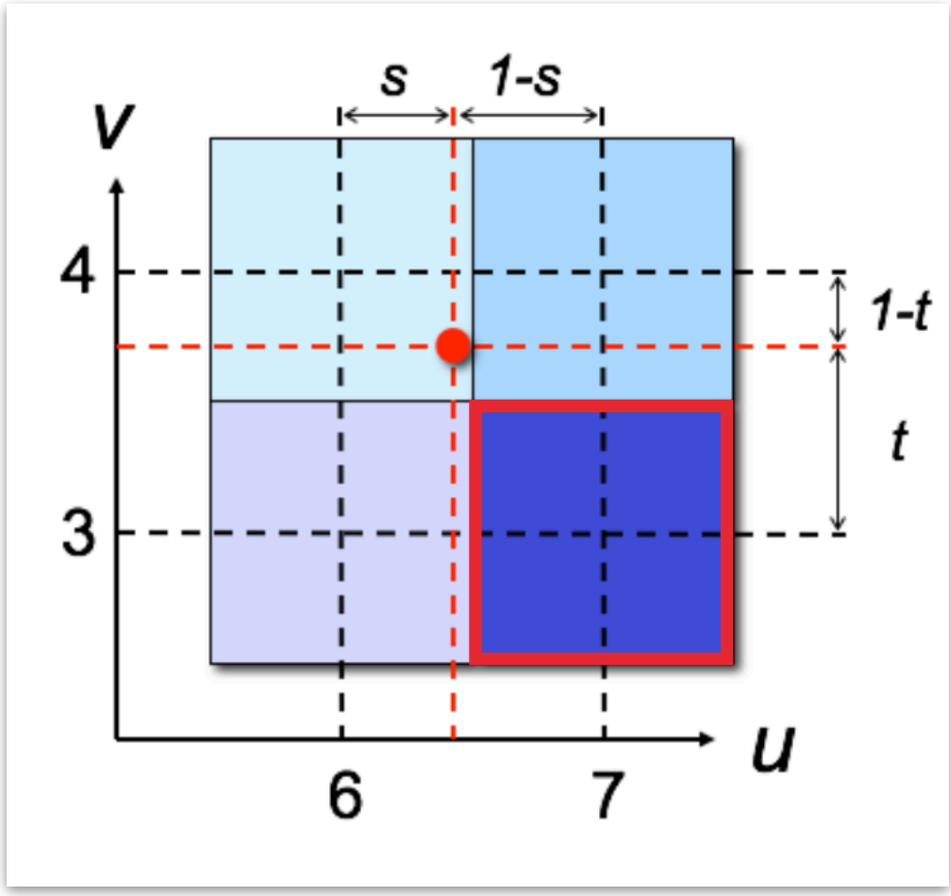



*[3]: Introduction to Computer Graphics (Spring 2020), EPFL, Prof. Mark Pauly, Lecture 7a: Texturing, Slide 20*

# BACKGROUND: BILINEAR FILTERING

- Texture coordinates: (u,v) pair defines a texel coordinate.
- Bilinear filtering: Sample texel and neighboring texels.

**Example 2x2 texture being sampled**

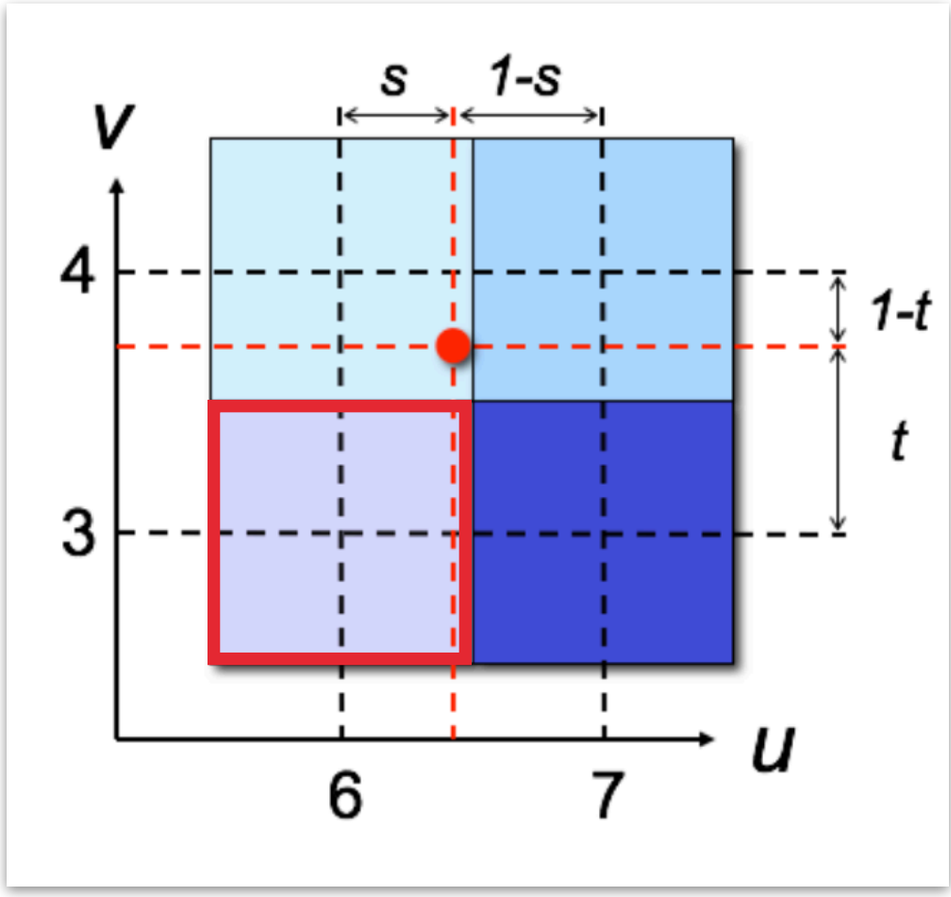



*[3]: Introduction to Computer Graphics (Spring 2020), EPFL, Prof. Mark Pauly, Lecture 7a: Texturing, Slide 20*

# BACKGROUND: BILINEAR FILTERING

- Texture coordinates: (u,v) pair defines a texel coordinate.
- Bilinear filtering: Sample texel and neighboring texels.

**Example 2x2 texture being sampled**

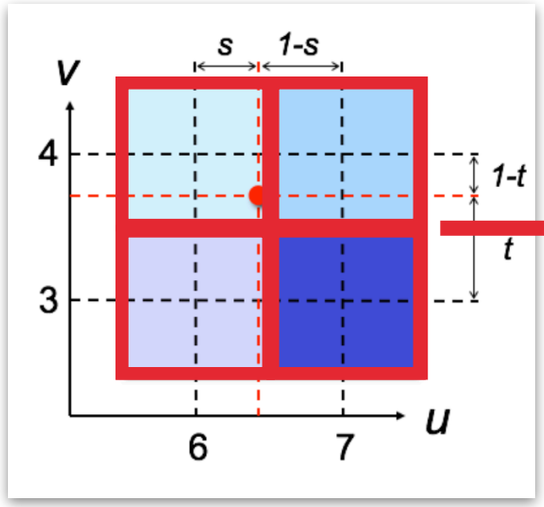



*[3]: Introduction to Computer Graphics (Spring 2020), EPFL, Prof. Mark Pauly, Lecture 7a: Texturing, Slide 20*

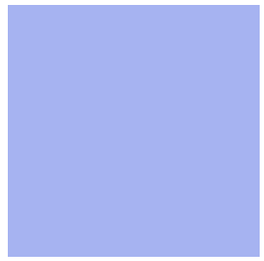
# BACKGROUND: BILINEAR FILTERING

- Texture coordinates: (u,v) pair defines a texel coordinate.
- Bilinear filtering: Sample texel and neighboring texels.
  - Result: Linear Interpolation between all sampled texels.

**Example 2x2 texture being sampled**




**Resulting Filtered Texel**

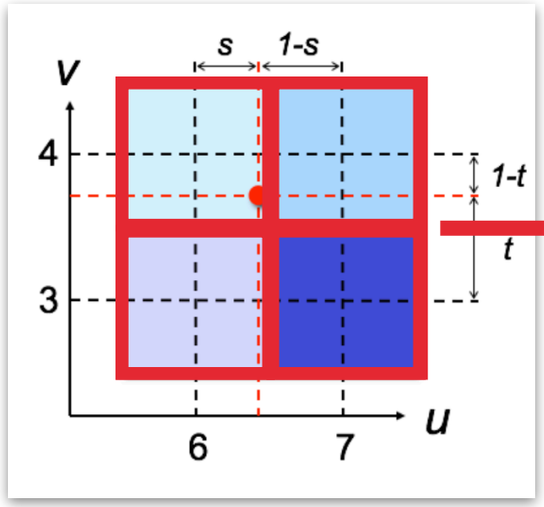


*[3]: Introduction to Computer Graphics (Spring 2020), EPFL, Prof. Mark Pauly, Lecture 7a: Texturing, Slide 20*

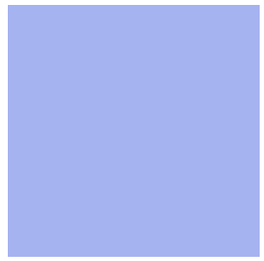
# BACKGROUND: BILINEAR FILTERING

- Texture coordinates: (u,v) pair defines a texel coordinate.
- Bilinear filtering: Sample texel and neighboring texels.
  - Result: Linear Interpolation between all sampled texels.
- Every sample requires **at least 4 memory requests**.

**Example 2x2 texture being sampled**




**Resulting Filtered Texel**



*[3]: Introduction to Computer Graphics (Spring 2020), EPFL, Prof. Mark Pauly, Lecture 7a: Texturing, Slide 20*

# BACKGROUND: TRILINEAR & ANISOTROPIC FILTERING

- **Trilinear Filtering**: “Multi-resolution” bilinear filtering
  - Sample pixels from the texture scaled in multiple ways.

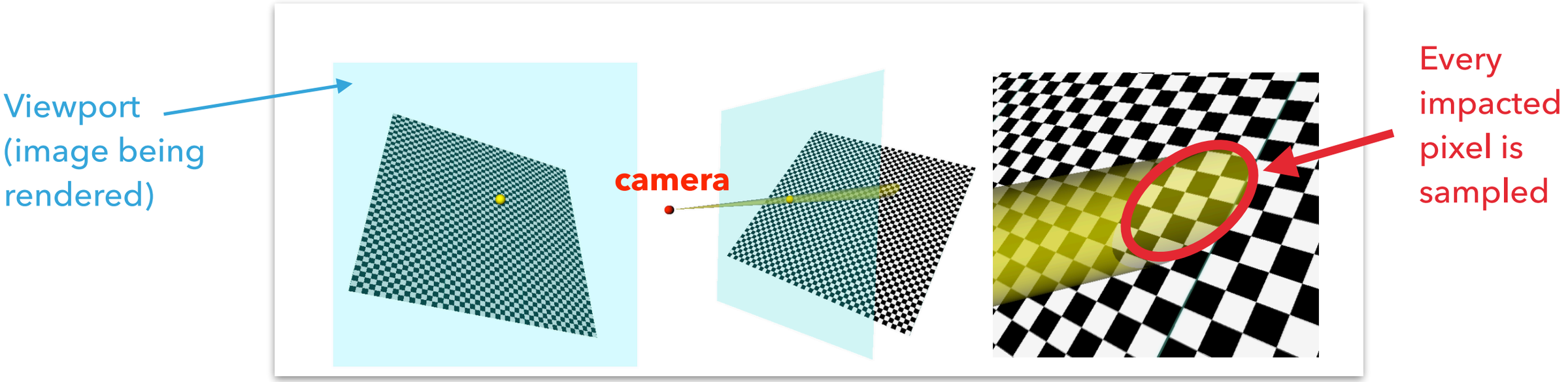
# BACKGROUND: TRILINEAR & ANISOTROPIC FILTERING

- **Trilinear Filtering**: “Multi-resolution” bilinear filtering
- **Anisotropic Filtering**: “Multi-angle” trilinear filtering
  - Sample pixels from the texture adapted for different camera angles.

# BACKGROUND: TRILINEAR & ANISOTROPIC FILTERING

- **Trilinear Filtering**: “Multi-resolution” bilinear filtering
- **Anisotropic Filtering**: “Multi-angle” trilinear filtering
  - Sample pixels from the texture adapted for different camera angles.

**Idea behind Anisotropic Filtering**

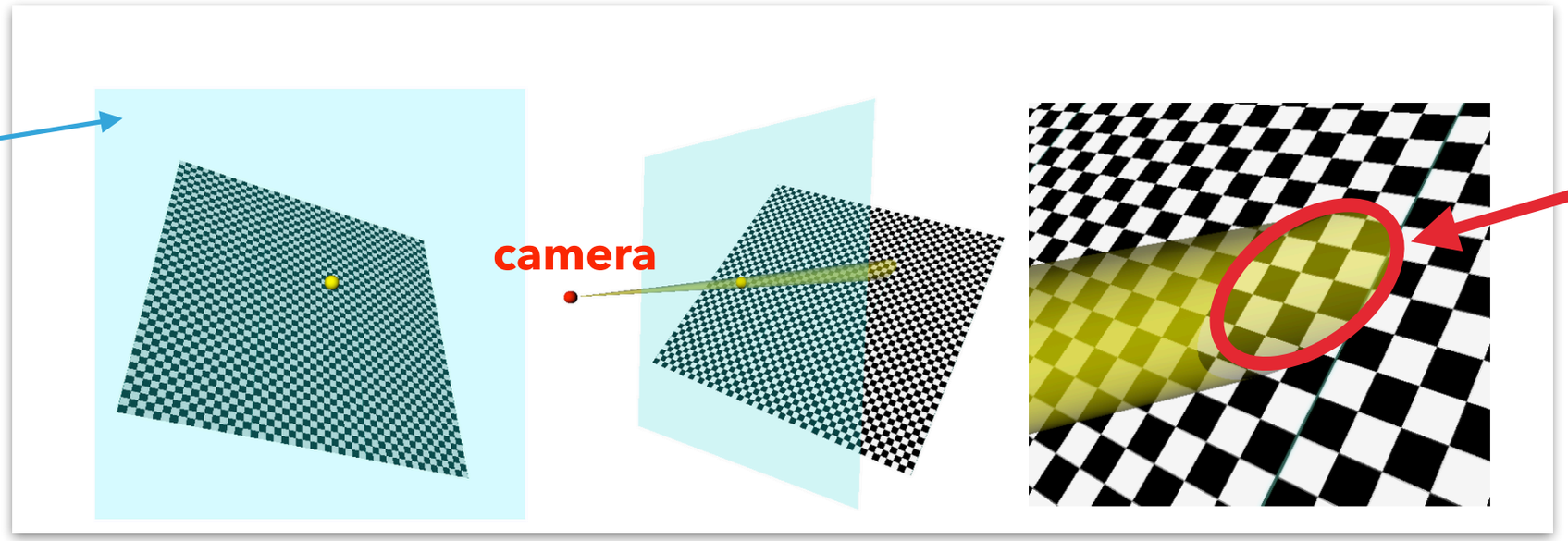



*[3]: Introduction to Computer Graphics (Spring 2020), EPFL, Prof. Mark Pauly, Lecture 7a: Texturing, Slide 25*

# BACKGROUND: TRILINEAR & ANISOTROPIC FILTERING

- **Trilinear Filtering**: "Multi-resolution" bilinear filtering
- **Anisotropic Filtering**: "Multi-angle" trilinear filtering

**Idea behind Anisotropic Filtering**



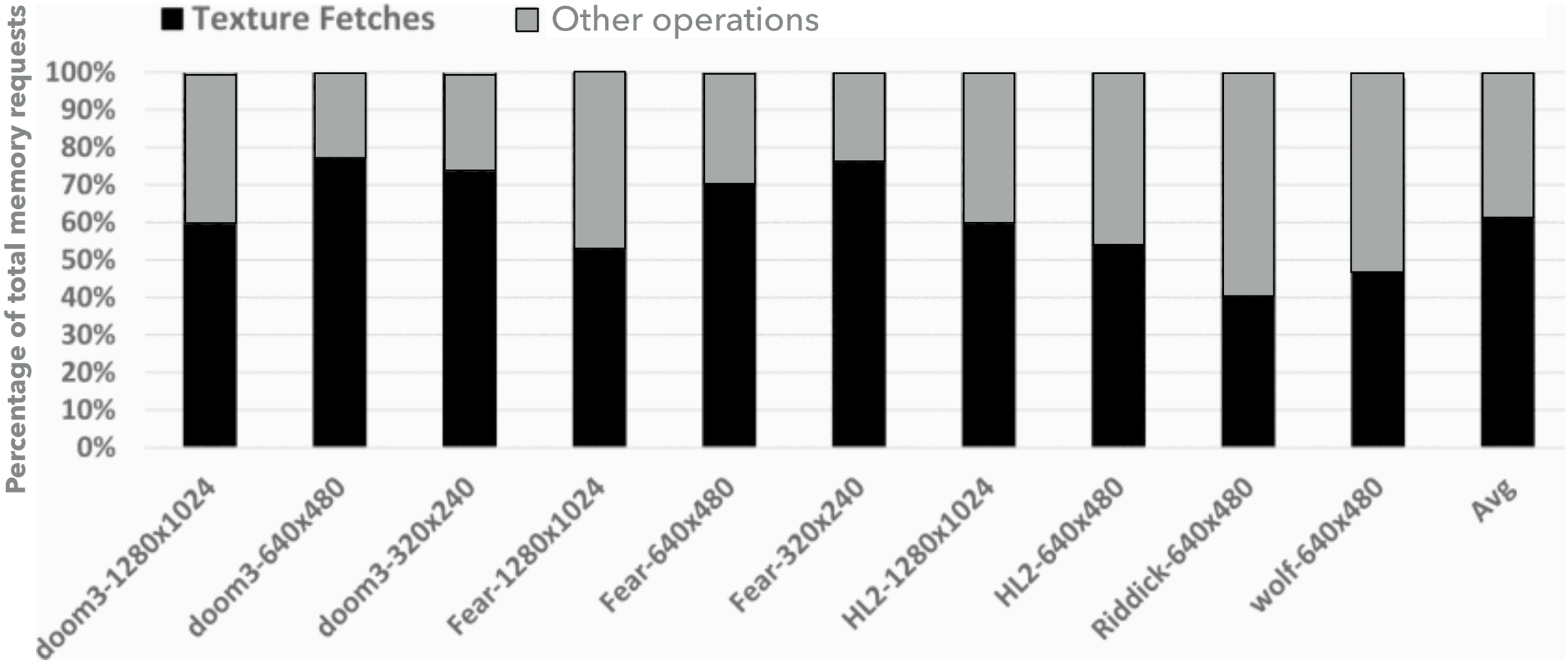

*[3]: Introduction to Computer Graphics (Spring 2020), EPFL, Prof. Mark Pauly, Lecture 7a: Texturing, Slide 25*

- Ex: **16x anisotropic filtering:**
  - 16x2x4 = 128 texels ==> **32x bilinear filtering**.

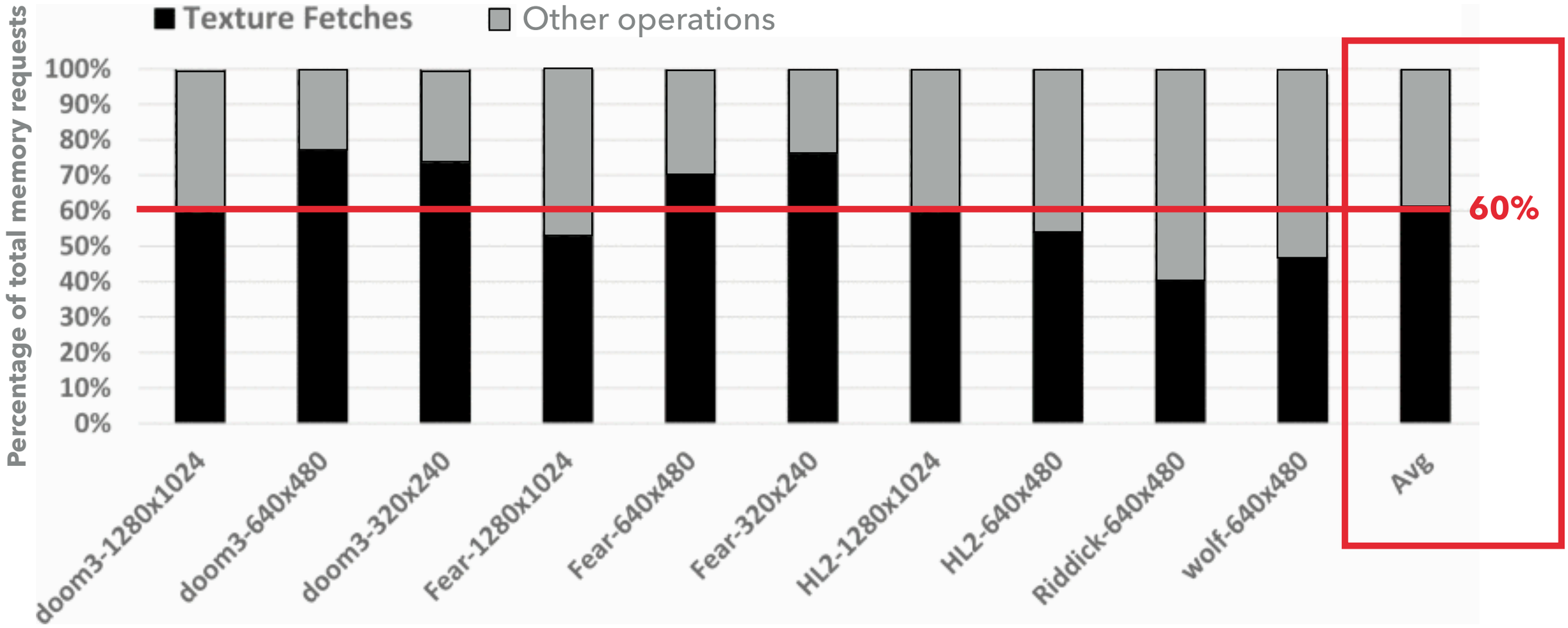
# OUTLINE

- Background: Texture Filtering
- Motivation
- Solutions:
  - B-PIM: Basic Processing In Memory
  - S-TFIM: Simple Texture Filtering In Memory
  - A-TFIM: Advanced Texture Filtering In Memory
- Evaluation Methodology
- Evaluation Results
- Conclusion

# MOTIVATION

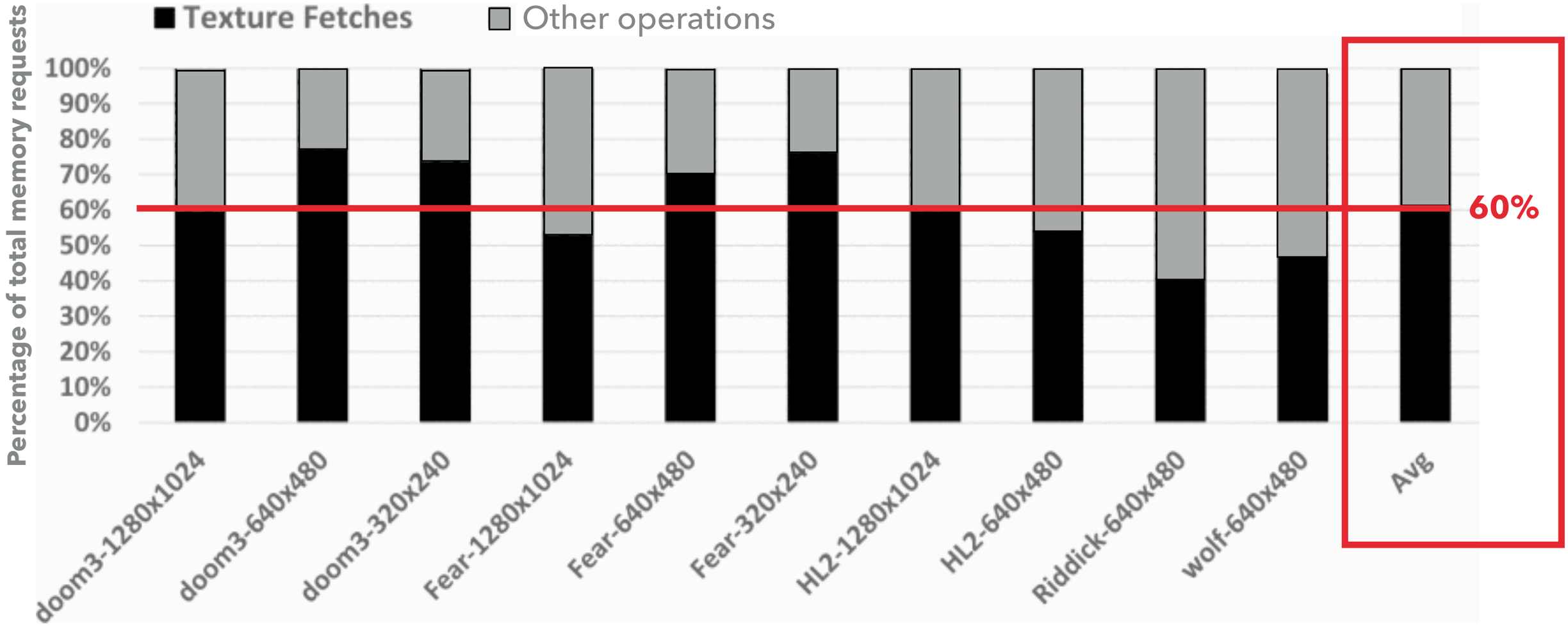

Texture Fetches
Other operations
Percentage of total memory requests
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
doom3-1280x1024
doom3-640x480
doom3-320x240
Fear-1280x1024
Fear-640x480
Fear-320x240
HL2-1280x1024
HL2-640x480
Riddick-640x480
wolf-640x480
Avg

# MOTIVATION



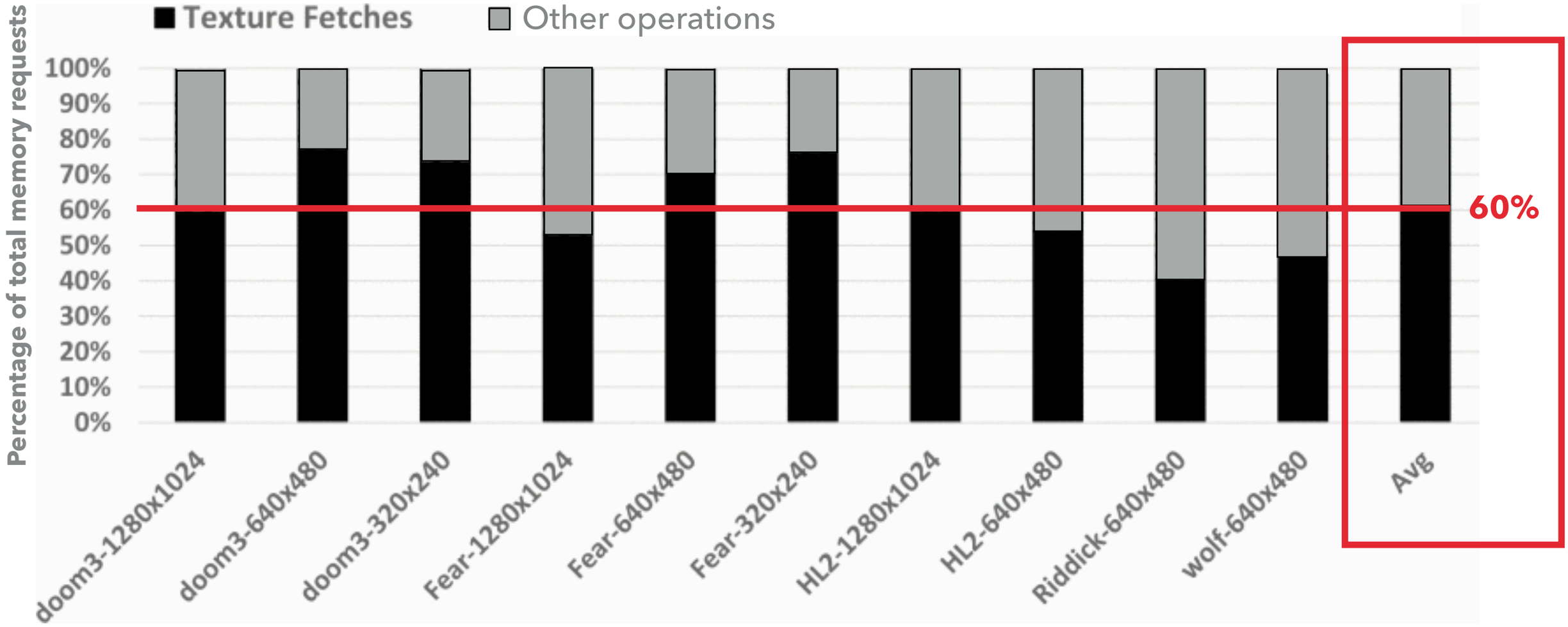

- **3D rendering**: **60%** of memory requests are texture fetches.

# MOTIVATION




- **3D rendering**: **60%** of memory requests are texture fetches.
- **Texture filtering causes memory to be a bottleneck in 3D rendering.**

# MOTIVATION




- **3D rendering**: **60%** of memory requests are texture fetches.
- **Texture filtering causes memory to be a bottleneck in 3D rendering.**
- **Solution: Texture Filtering In Memory**

# OUTLINE

- Background: Texture Filtering
- Motivation
- Solutions:
  - B-PIM: Basic Processing In Memory
  - S-TFIM: Simple Texture Filtering In Memory
  - A-TFIM: Advanced Texture Filtering In Memory
- Evaluation Methodology
- Evaluation Results
- Conclusion

# B-PIM: BASIC PROCESSING-IN-MEMORY

- Idea: Replace GPU memory with Hybrid Memory Cube.

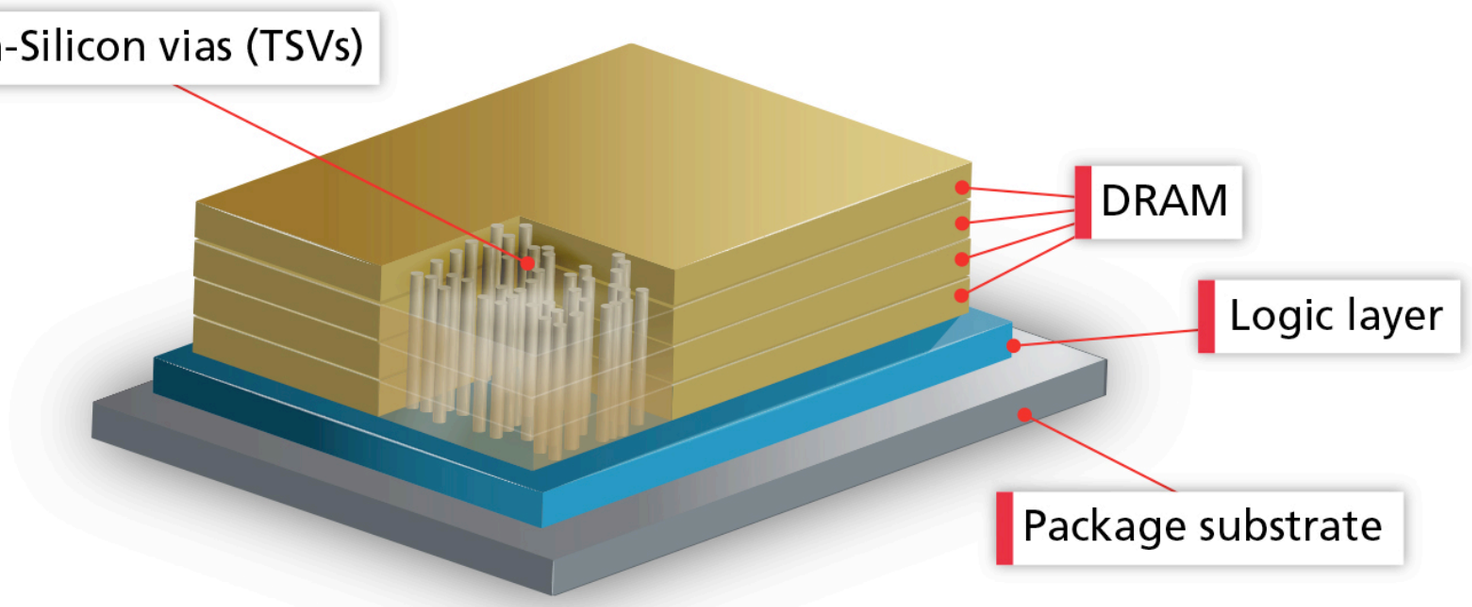
# B-PIM: BASIC PROCESSING-IN-MEMORY

- Idea: Replace GPU memory with Hybrid Memory Cube.
- **Hybrid Memory Cube**: High bandwidth 3D stacked DRAM technology with a logic layer at the base.

# B-PIM: BASIC PROCESSING-IN-MEMORY

- Idea: Replace GPU memory with Hybrid Memory Cube.
- **Hybrid Memory Cube**: High bandwidth 3D stacked DRAM technology with a logic layer at the base.
- Vertical connection = **Through-Silicon Via** (TSV):
  - **Internal** bandwidth (connection to logic layer):
    - **512 GB/s**
  - **External** bandwidth (connection to host):
    - **320 GB/s**

*[2]: https://community.cadence.com/cadence_blogs_8/b/fv/posts/what-s-new-with-hybrid-memory-cube-hmc*




HMC Memory Chip Architecture

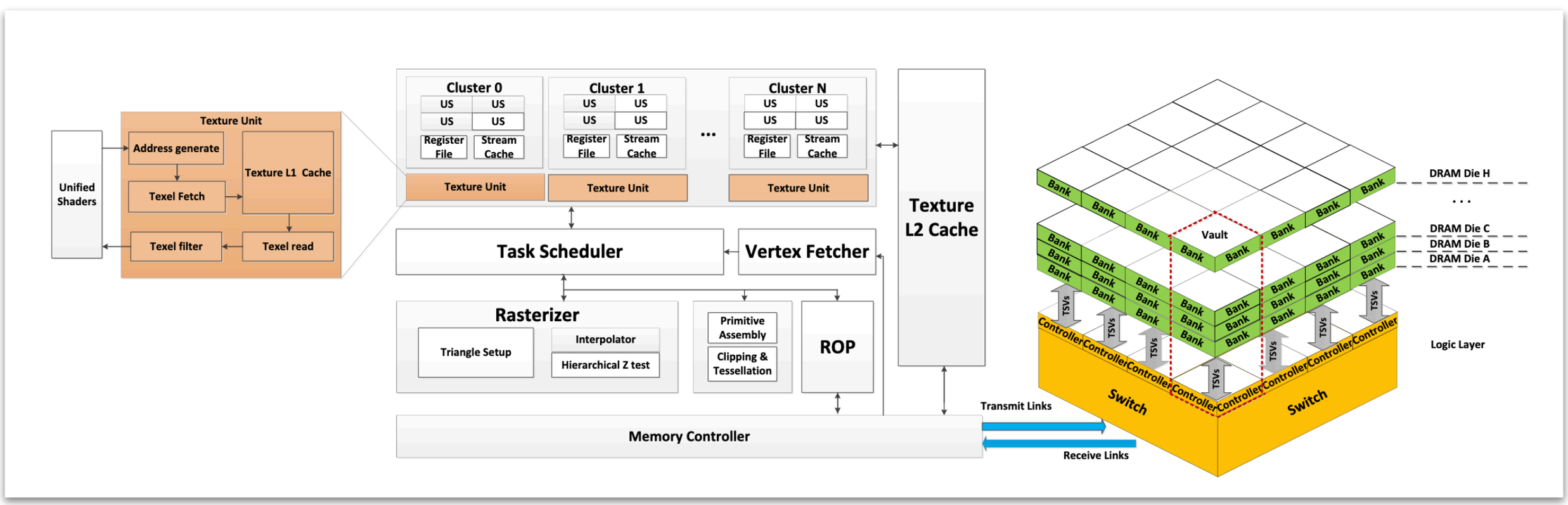
# B-PIM: BASIC PROCESSING-IN-MEMORY

- Overview: **GPU is the same**, performance is increased by having **HMC as memory**.

# B-PIM: BASIC PROCESSING-IN-MEMORY

- Overview: **GPU is the same**, performance is increased by having **HMC as memory**.

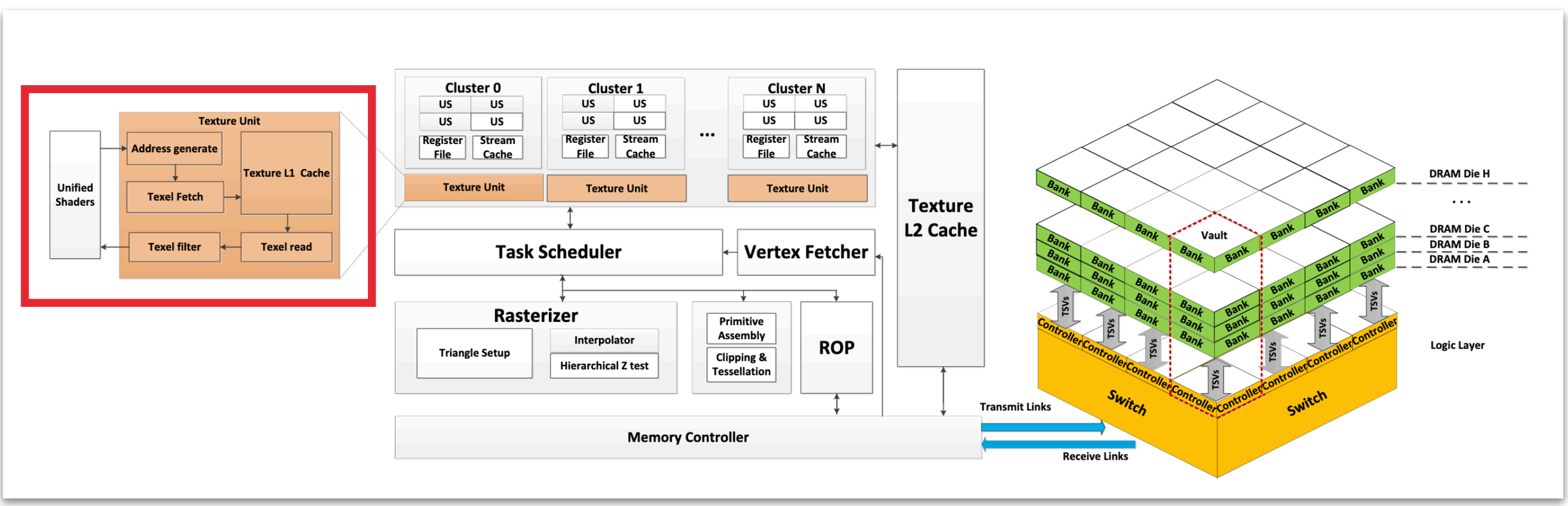
**Overview of the B-PIM baseline**



# B-PIM: BASIC PROCESSING-IN-MEMORY

- Overview: **GPU is the same**, performance is increased by having **HMC as memory**.

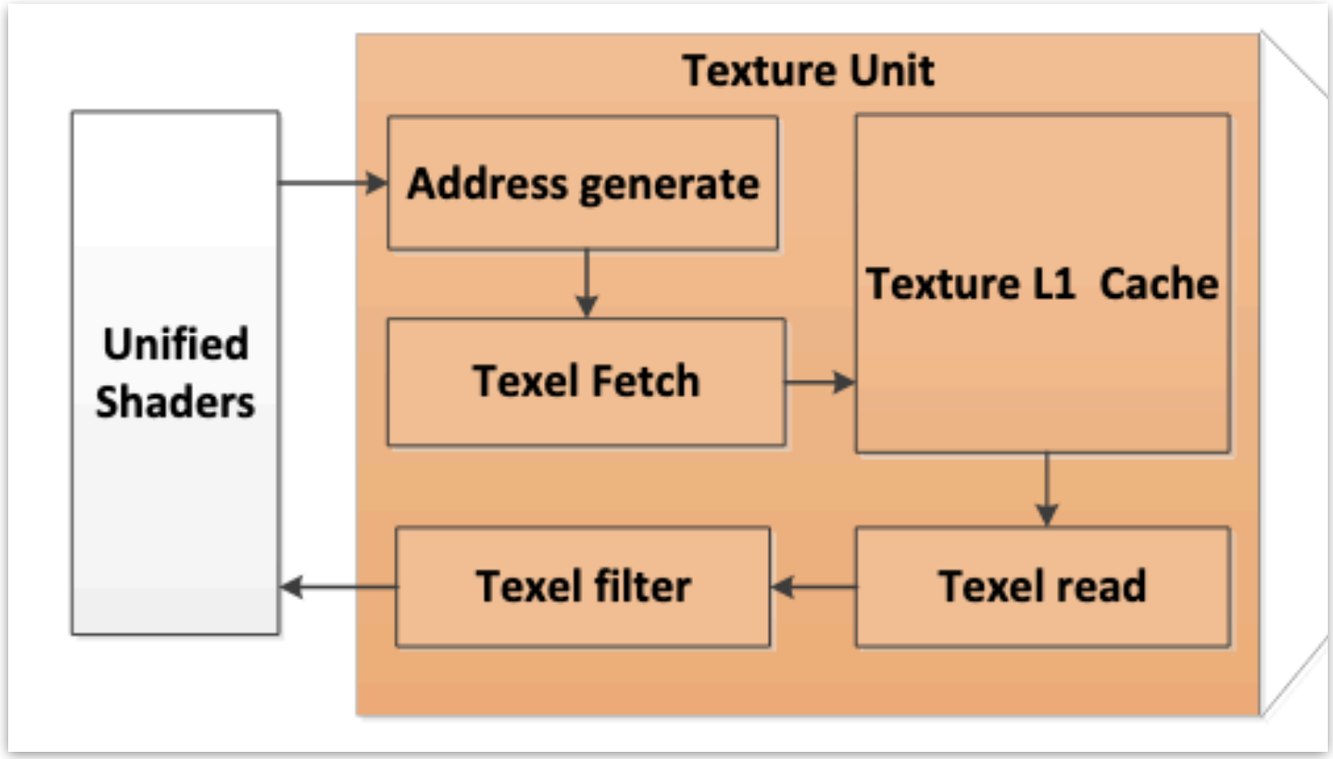
**Overview of the B-PIM baseline**



# B-PIM: BASIC PROCESSING-IN-MEMORY

- Overview: **GPU is the same**, performance is increased by having **HMC as memory**.

**Overview of the B-PIM baseline Texture Unit**




- **Problem**: Doesn't exploit internal bandwidth of HMC.

# OUTLINE

- Background: Texture Filtering
- Motivation
- Solutions:
  - B-PIM: Basic Processing In Memory
  - S-TFIM: Simple Texture Filtering In Memory
  - A-TFIM: Advanced Texture Filtering In Memory
- Evaluation Methodology
- Evaluation Results
- Conclusion

# S-TFIM: SIMPLE TEXTURE FILTERING IN MEMORY

- Idea: Move **all texture filtering** to logic layer of HMC.

# S-TFIM: SIMPLE TEXTURE FILTERING IN MEMORY

- Idea: Move **all texture filtering** to logic layer of HMC.
- Unified Shader (US) requests texel => HMC provides filtered texel.

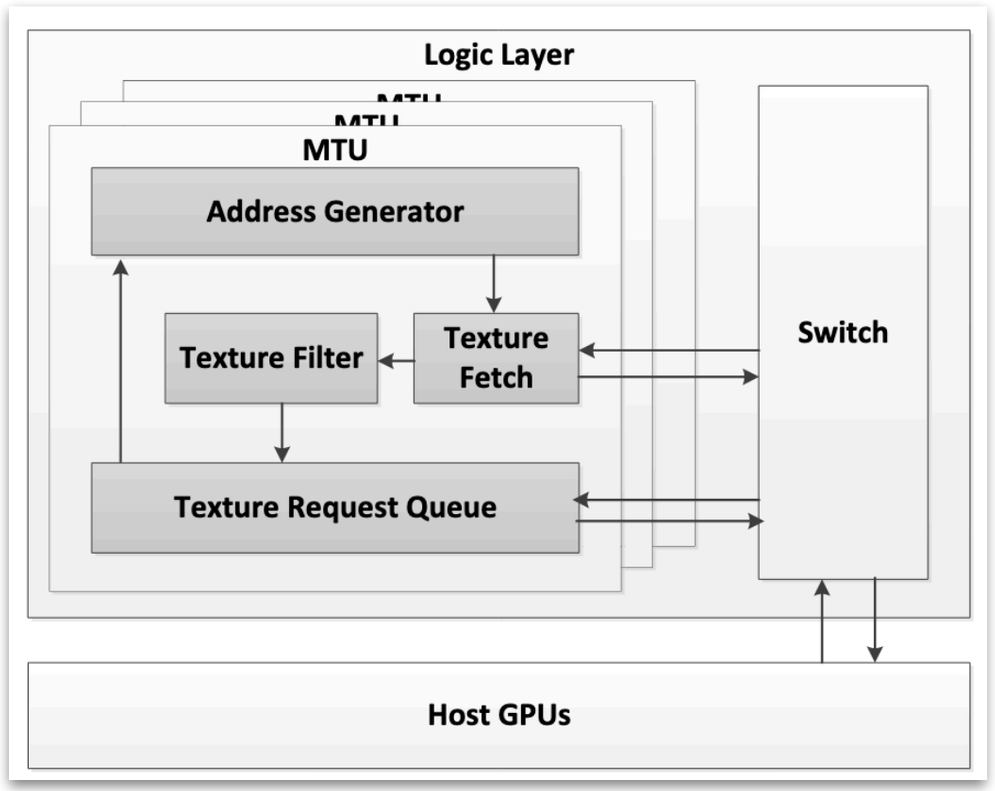
# S-TFIM: SIMPLE TEXTURE FILTERING IN MEMORY

- Idea: Move **all texture filtering** to logic layer of HMC.
- Unified Shader (US) requests texel => HMC provides filtered texel.
- Memory Texture Unit (MTU): Does filtering in Logic layer of HMC.

# S-TFIM: SIMPLE TEXTURE FILTERING IN MEMORY

- Idea: Move **all texture filtering** to logic layer of HMC.
- Unified Shader (US) requests texel => HMC provides filtered texel.
- Memory Texture Unit (MTU): Does filtering in Logic layer of HMC.

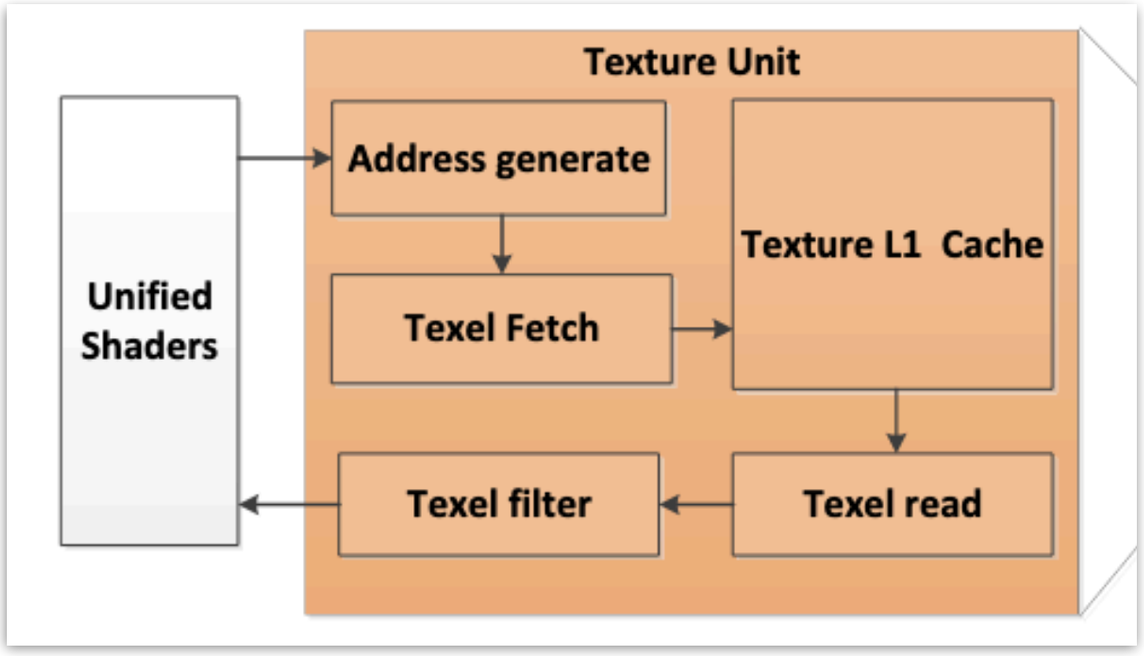
## Overview of S-TFIM architecture



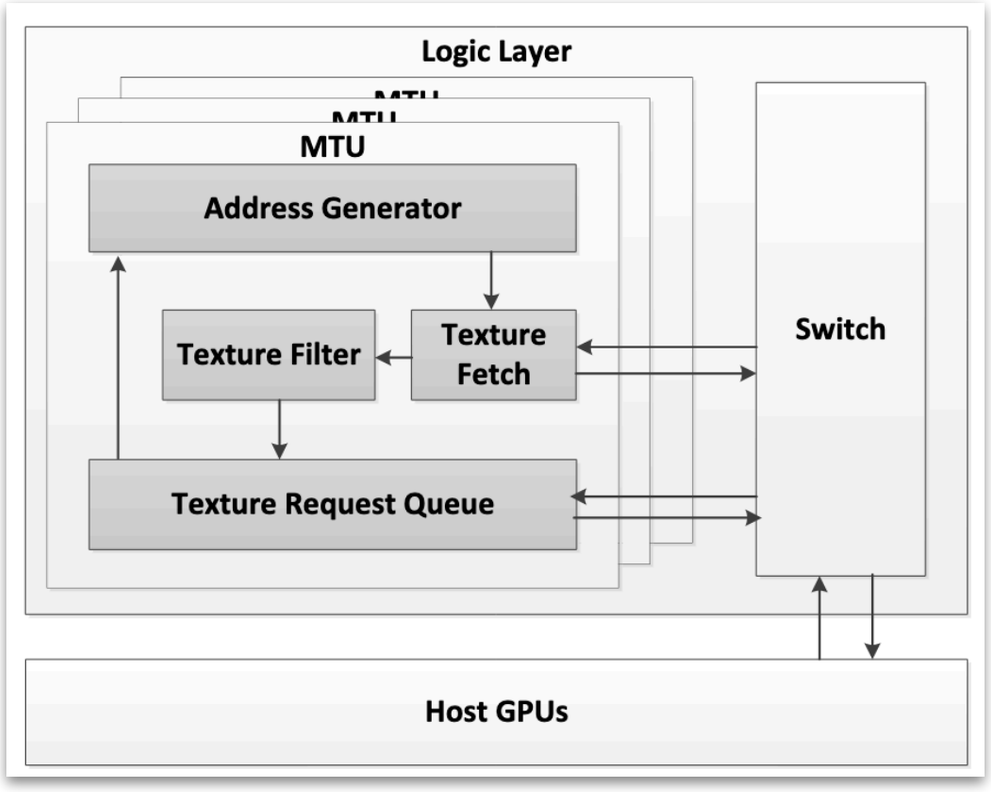
# S-TFIM: SIMPLE TEXTURE FILTERING IN MEMORY

- Idea: Move **all texture filtering** to logic layer of HMC.
- Unified Shader (US) requests texel => HMC provides filtered texel.
- Memory Texture Unit (MTU): Does filtering in Logic layer of HMC.

**Overview of the B-PIM baseline Texture Unit**



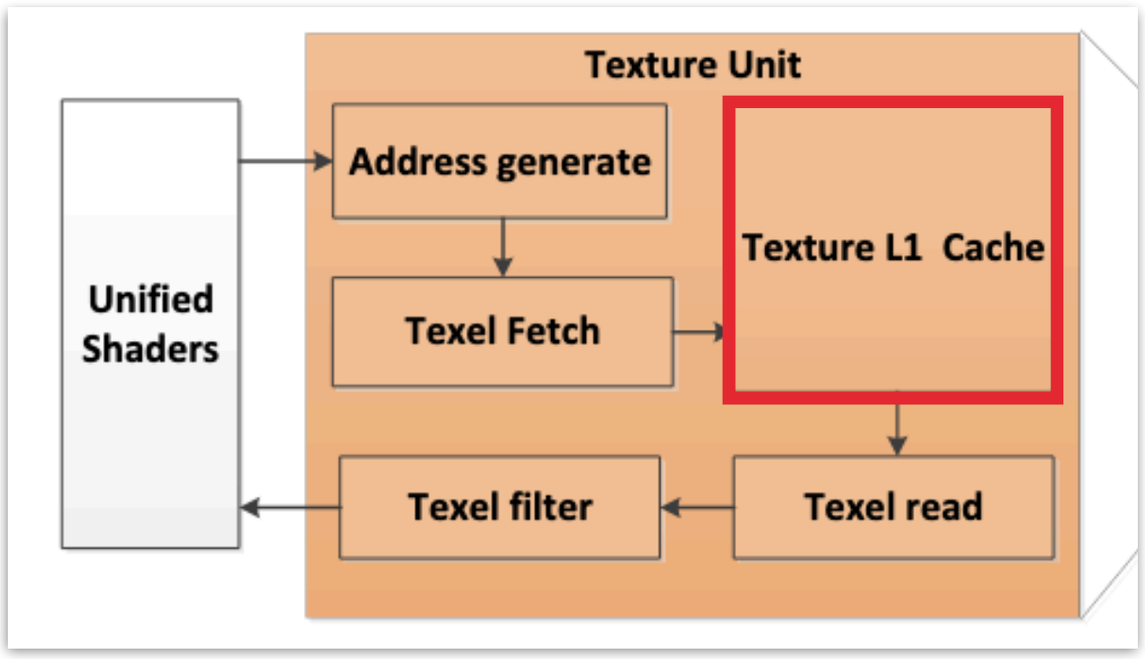

**Overview of S-TFIM architecture**



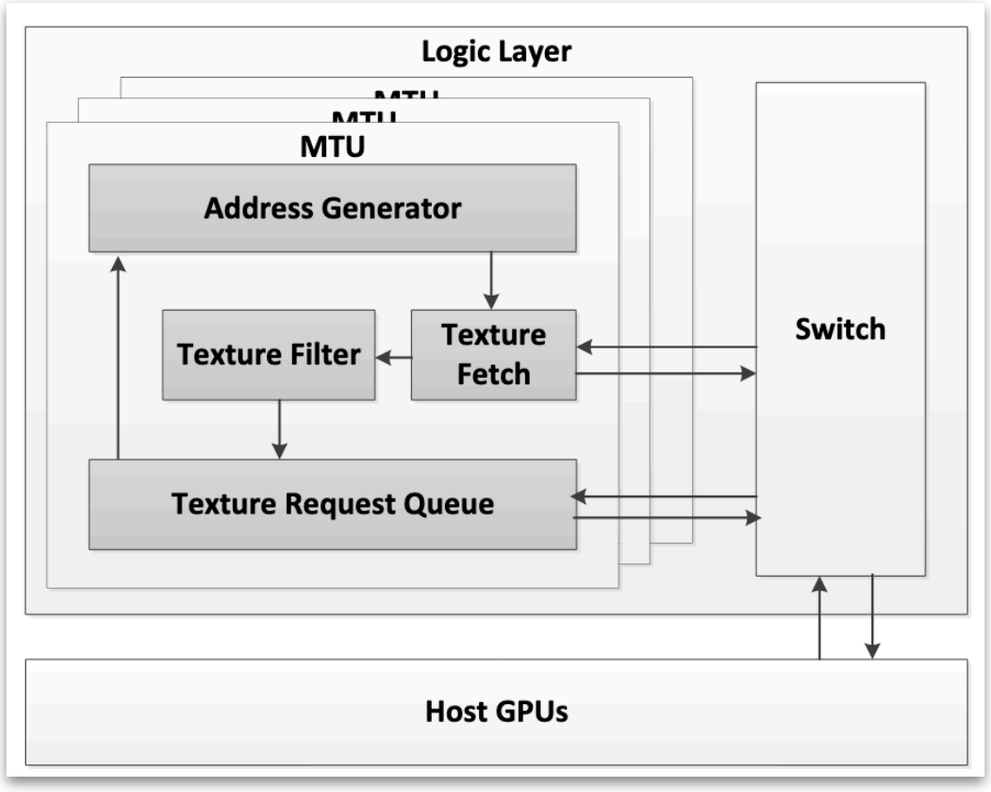
# S-TFIM: SIMPLE TEXTURE FILTERING IN MEMORY

- Idea: Move **all texture filtering** to logic layer of HMC.
- Unified Shader (US) requests texel => HMC provides filtered texel.
- Memory Texture Unit (MTU): Does filtering in Logic layer of HMC.

**Overview of the B-PIM baseline Texture Unit**




**Overview of S-TFIM architecture**




- **Problem**: Intermediate filtered-texel caching is no longer possible.

# OUTLINE

- Background: Texture Filtering
- Motivation
- Solutions:
  - B-PIM: Basic Processing In Memory
  - S-TFIM: Simple Texture Filtering In Memory
  - A-TFIM: Advanced Texture Filtering In Memory
- Evaluation Methodology
- Evaluation Results
- Conclusion

# A-TFIM: ADVANCED TEXTURE FILTERING IN MEMORY

- Idea: Focus on **most memory hungry phase** ==> **Anisotropic Filtering**

# A-TFIM: ADVANCED TEXTURE FILTERING IN MEMORY

- Idea: Focus on **most memory hungry phase** ==> **Anisotropic Filtering**
- Two types of texels:

# A-TFIM: ADVANCED TEXTURE FILTERING IN MEMORY

- Idea: Focus on **most memory hungry phase** ==> **Anisotropic Filtering**
- Two types of texels:
  - **parent texels:** the requested texel.

# A-TFIM: ADVANCED TEXTURE FILTERING IN MEMORY

- Idea: Focus on **most memory hungry phase** ==> **Anisotropic Filtering**

- Two types of texels:
  - **parent texels:** the requested texel.
  - **children texels:** the texels neighboring the requested one.

# A-TFIM: ADVANCED TEXTURE FILTERING IN MEMORY

- Idea: Focus on **most memory hungry phase** ==> **Anisotropic Filtering**
- Two types of texels:
  - **parent texels:** the requested texel.
  - **children texels:** the texels neighboring the requested one.
- Children texels are fetched directly in memory ==> lower latency.

# A-TFIM: ADVANCED TEXTURE FILTERING IN MEMORY

- Idea: Only move **most memory hungry phase** to memory.

# A-TFIM: ADVANCED TEXTURE FILTERING IN MEMORY

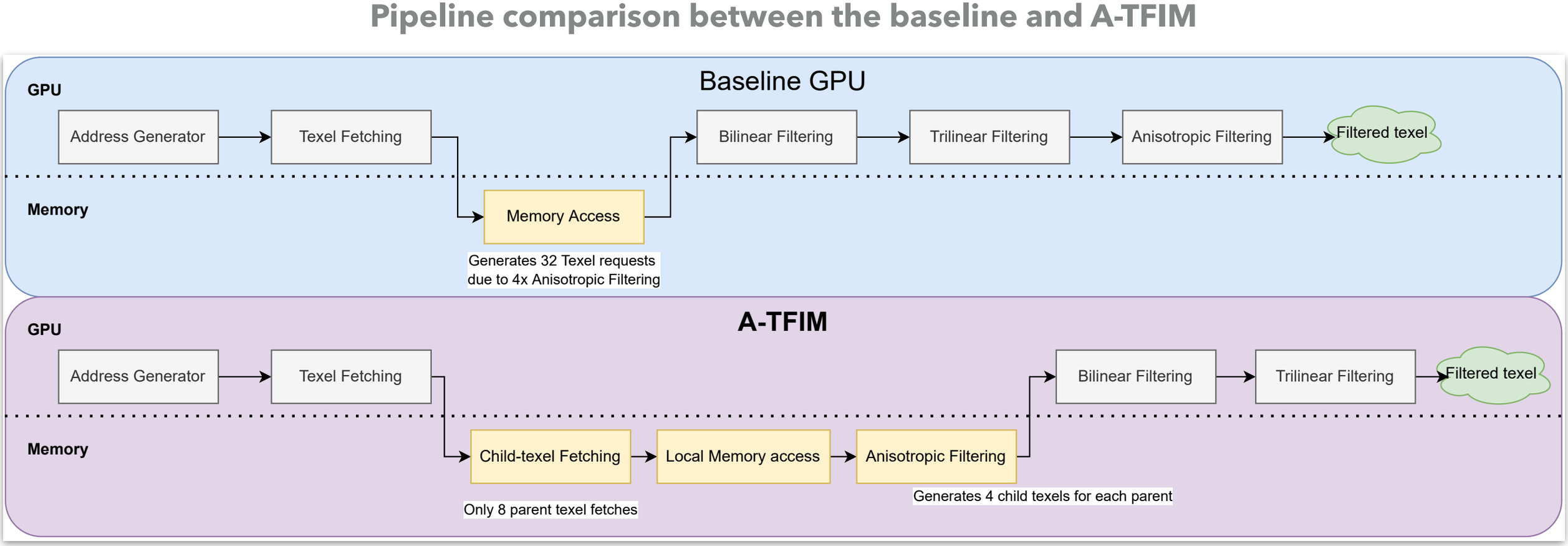
- Idea: Only move **most memory hungry phase** to memory.
- **Anisotropic Filtering** is done **before** the other phases.

# A-TFIM: ADVANCED TEXTURE FILTERING IN MEMORY

- Idea: Only move **most memory hungry phase** to memory.
- **Anisotropic Filtering** is done **before** the other phases.
- **Greatly reduces** memory traffic between host and HMC.

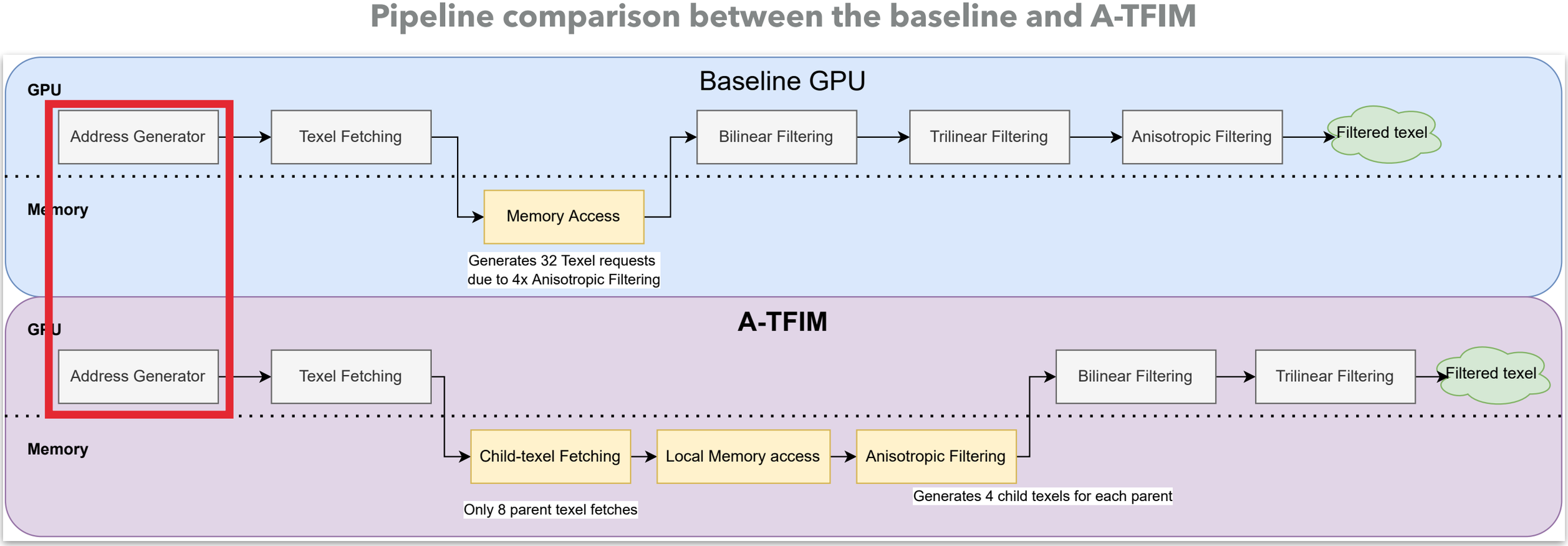
# A-TFIM: ADVANCED TEXTURE FILTERING IN MEMORY

- Idea: Only move **most memory hungry phase** to memory.
- **Anisotropic Filtering** is done **before** the other phases.
- **Greatly reduces** memory traffic between host and HMC.



# A-TFIM: ADVANCED TEXTURE FILTERING IN MEMORY

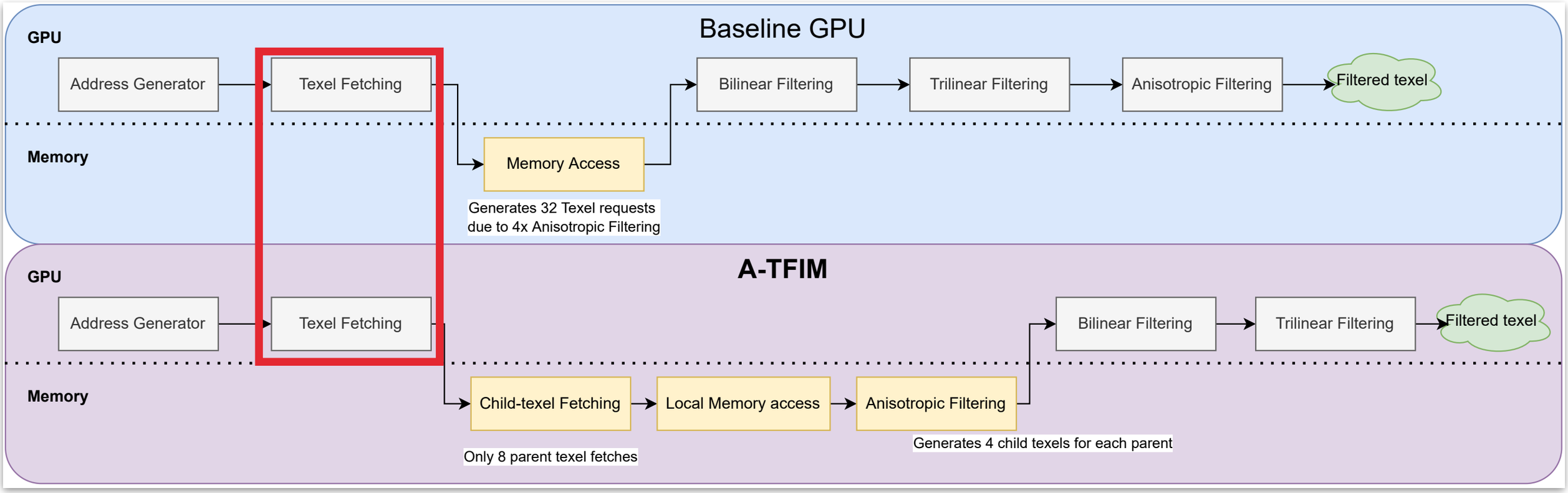
- Idea: Only move **most memory hungry phase** to memory.
- **Anisotropic Filtering** is done **before** the other phases.
- **Greatly reduces** memory traffic between host and HMC.



# A-TFIM: ADVANCED TEXTURE FILTERING IN MEMORY

- Idea: Only move **most memory hungry phase** to memory.
- **Anisotropic Filtering** is done **before** the other phases.
- **Greatly reduces** memory traffic between host and HMC.

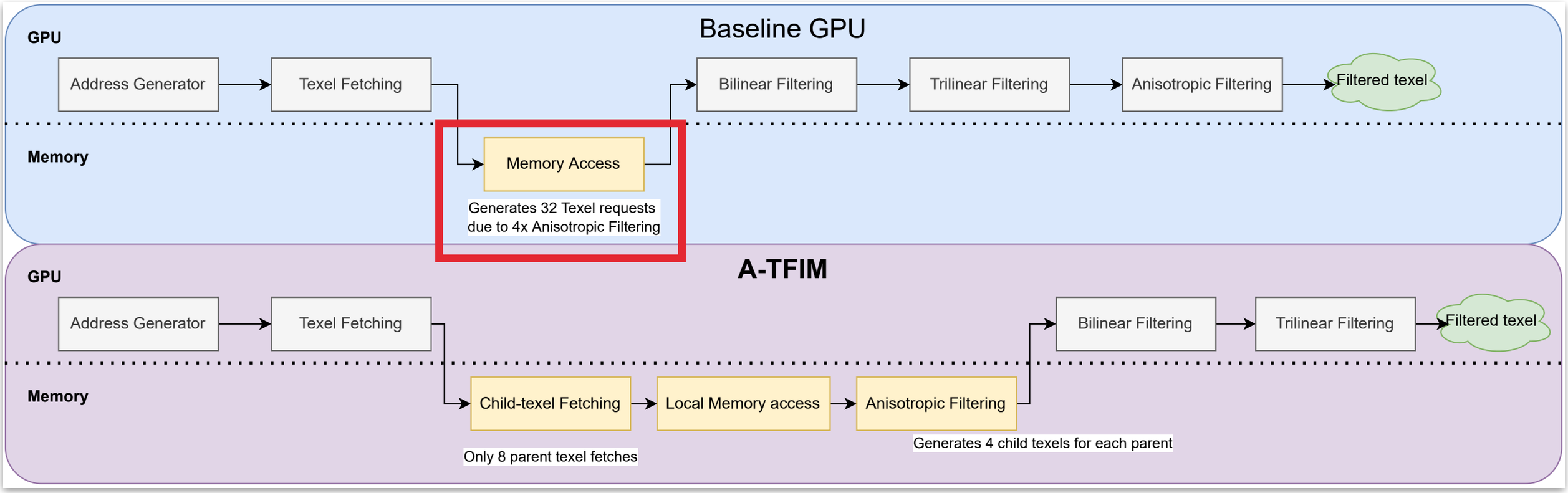
**Pipeline comparison between the baseline and A-TFIM**



# A-TFIM: ADVANCED TEXTURE FILTERING IN MEMORY

- Idea: Only move **most memory hungry phase** to memory.
- **Anisotropic Filtering** is done **before** the other phases.
- **Greatly reduces** memory traffic between host and HMC.

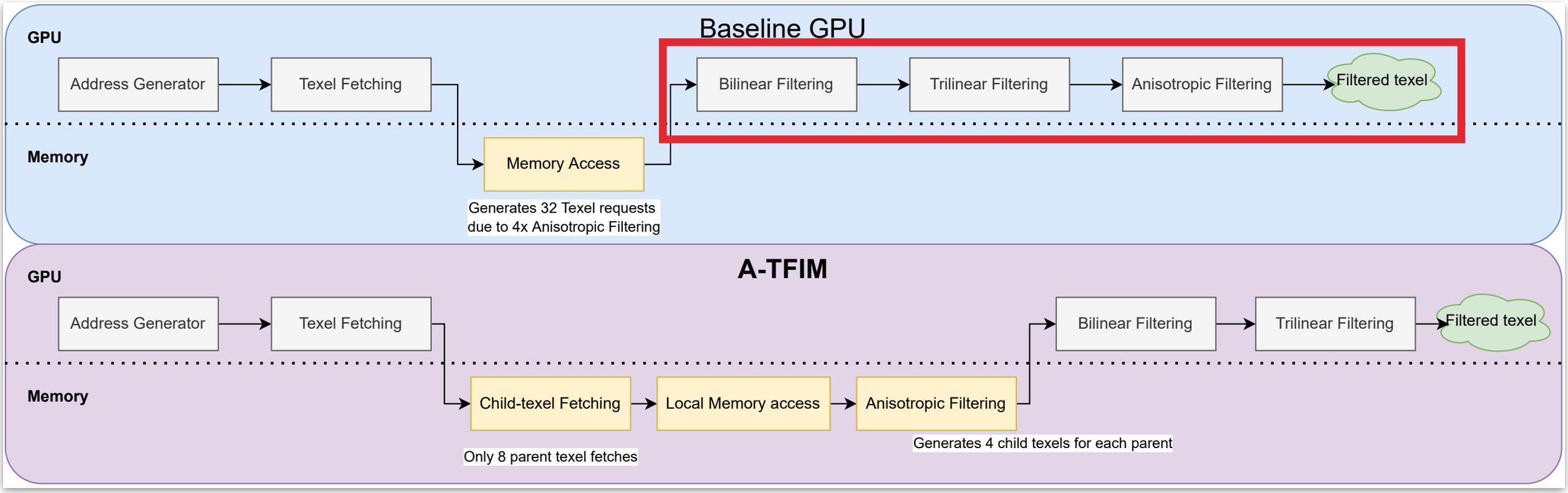
**Pipeline comparison between the baseline and A-TFIM**



# A-TFIM: ADVANCED TEXTURE FILTERING IN MEMORY

- Idea: Only move **most memory hungry phase** to memory.
- **Anisotropic Filtering** is done **before** the other phases.
- **Greatly reduces** memory traffic between host and HMC.

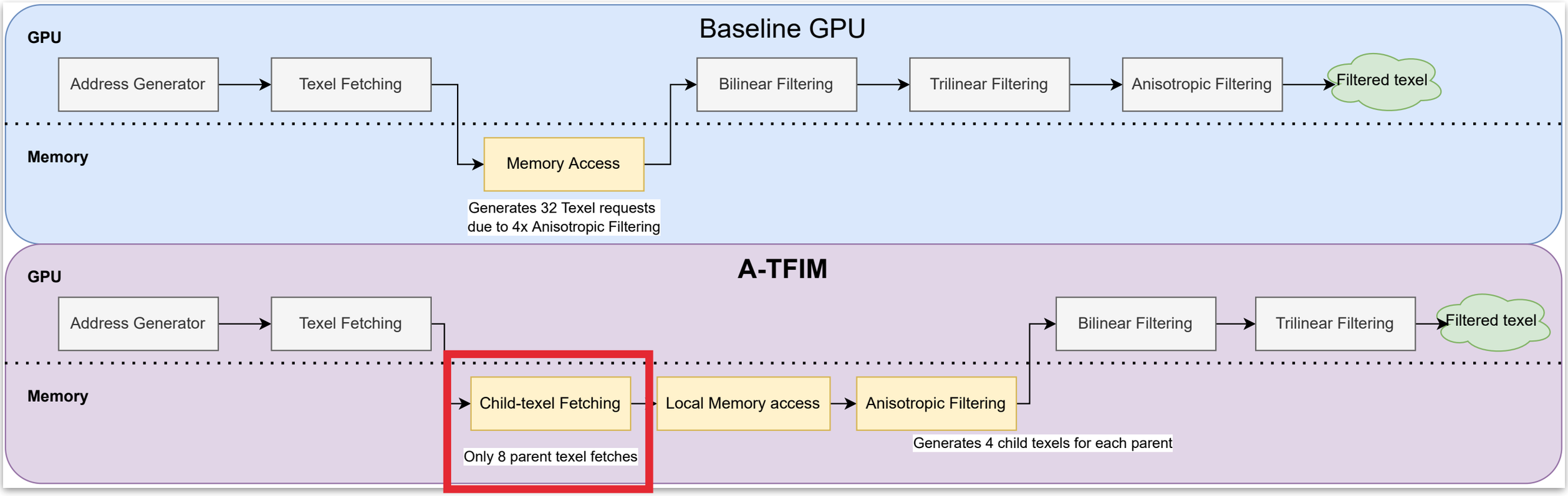
**Pipeline comparison between the baseline and A-TFIM**



# A-TFIM: ADVANCED TEXTURE FILTERING IN MEMORY

- Idea: Only move **most memory hungry phase** to memory.
- **Anisotropic Filtering** is done **before** the other phases.
- **Greatly reduces** memory traffic between host and HMC.

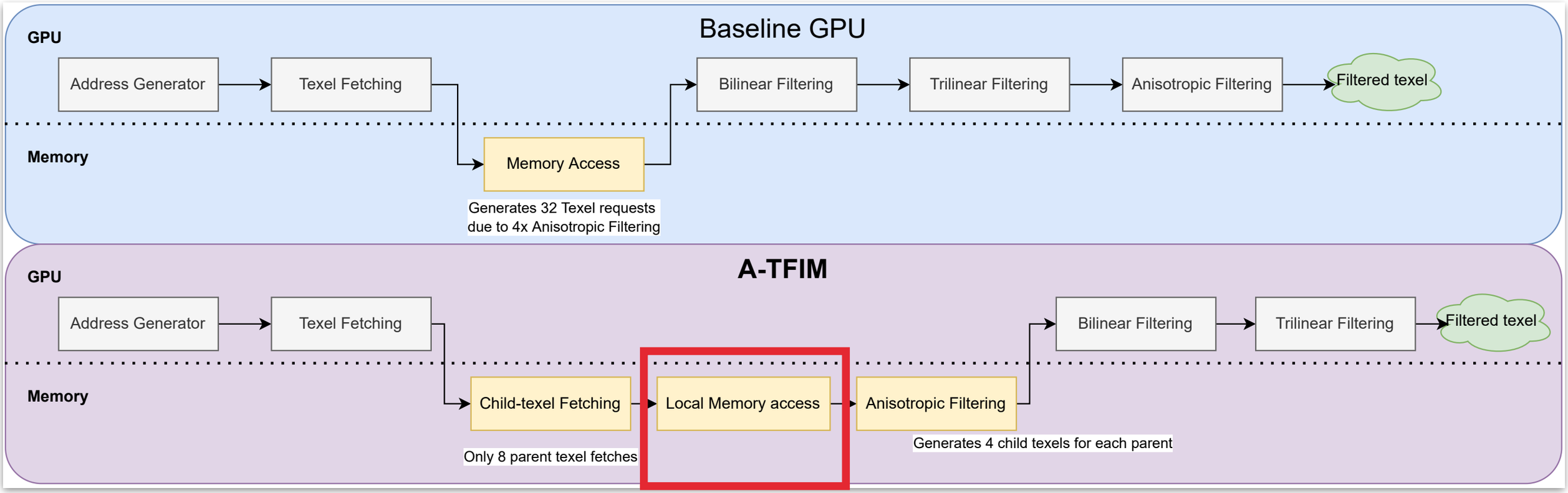
**Pipeline comparison between the baseline and A-TFIM**



# A-TFIM: ADVANCED TEXTURE FILTERING IN MEMORY

- Idea: Only move **most memory hungry phase** to memory.
- **Anisotropic Filtering** is done **before** the other phases.
- **Greatly reduces** memory traffic between host and HMC.

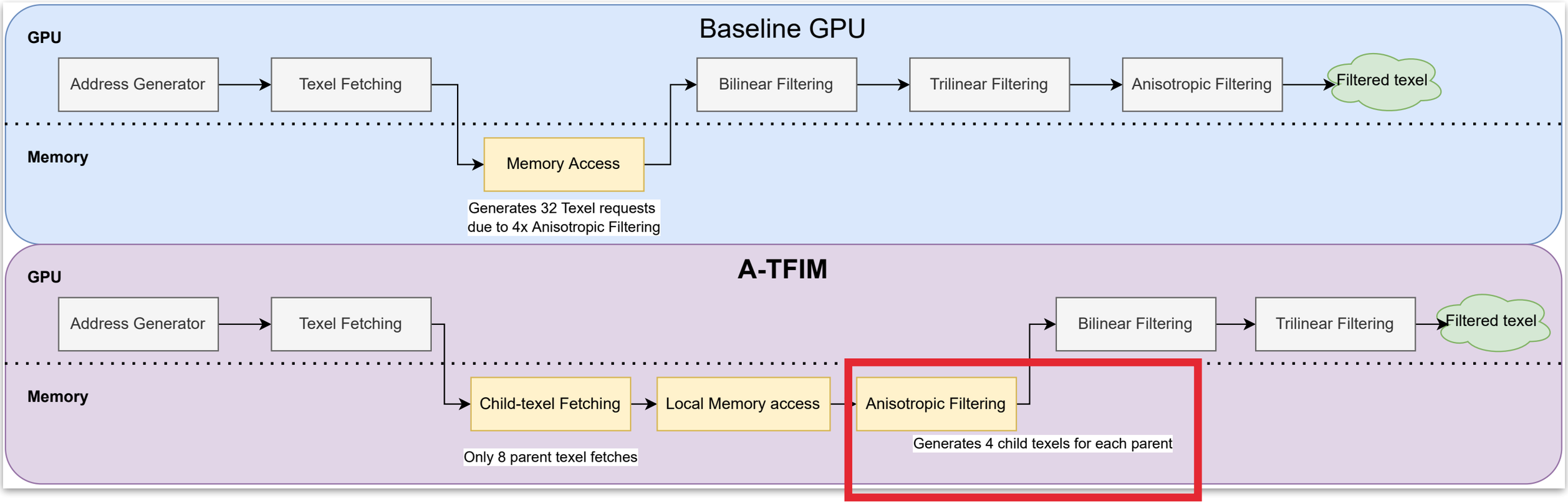
**Pipeline comparison between the baseline and A-TFIM**



# A-TFIM: ADVANCED TEXTURE FILTERING IN MEMORY

- Idea: Only move **most memory hungry phase** to memory.
- **Anisotropic Filtering** is done **before** the other phases.
- **Greatly reduces** memory traffic between host and HMC.

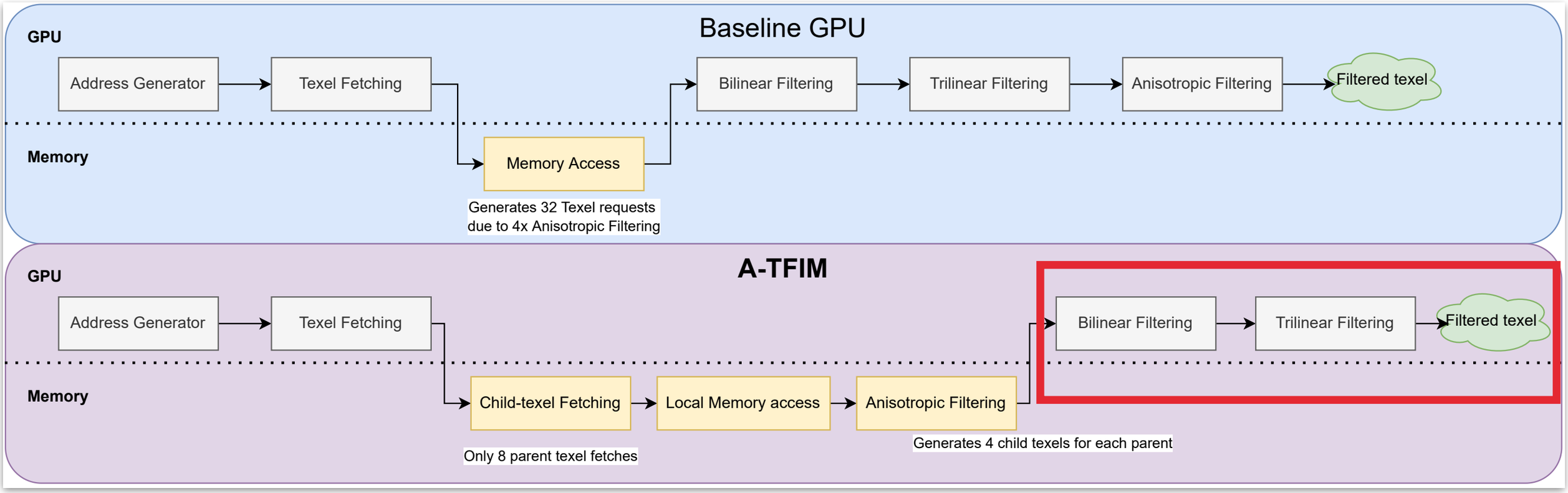
**Pipeline comparison between the baseline and A-TFIM**



# A-TFIM: ADVANCED TEXTURE FILTERING IN MEMORY

- Idea: Only move **most memory hungry phase** to memory.
- **Anisotropic Filtering** is done **before** the other phases.
- **Greatly reduces** memory traffic between host and HMC.

**Pipeline comparison between the baseline and A-TFIM**




- **A-TFIM allows intermediate filtered texels to be cached like in baseline.**

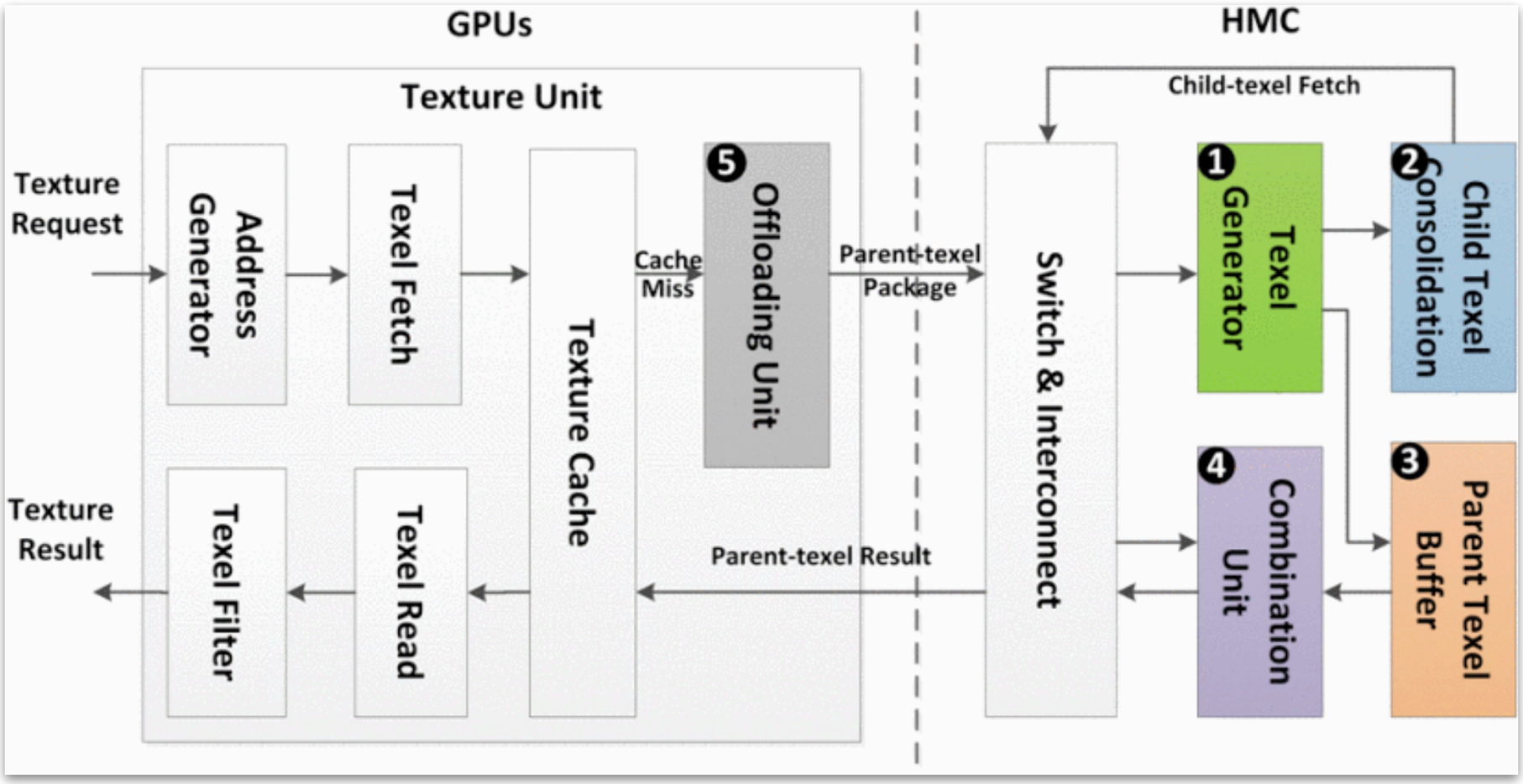
# A-TFIM: ADVANCED TEXTURE FILTERING IN MEMORY

- A-TFIM: Allows for texel caching.
  - Reduction in memory traffic caused by forced re-computation.

# A-TFIM: ADVANCED TEXTURE FILTERING IN MEMORY

- A-TFIM: Allows for texel caching.
  - Reduction in memory traffic caused by forced re-computation.

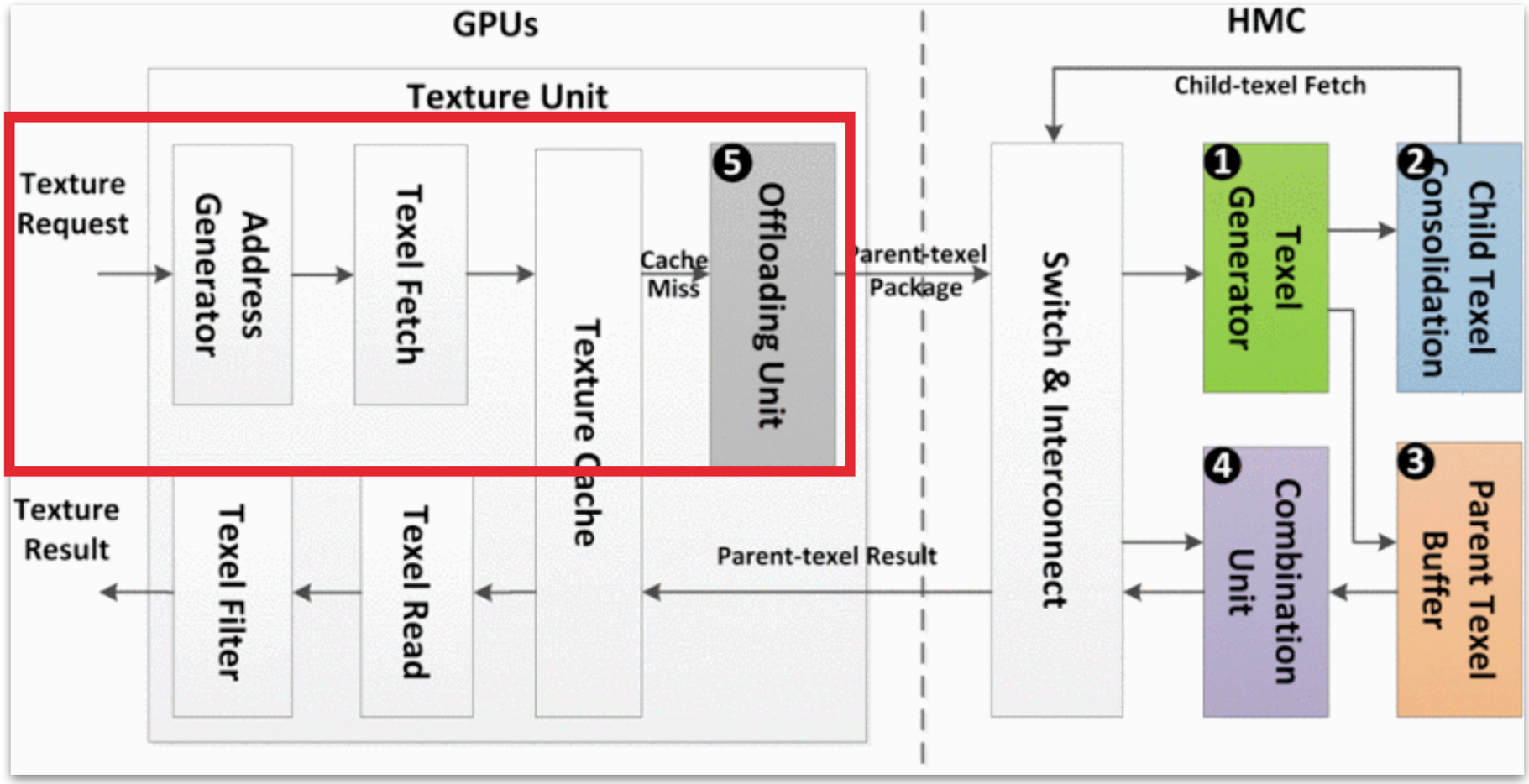
**Overview of A-TFIM Architecture**



# A-TFIM: ADVANCED TEXTURE FILTERING IN MEMORY

- A-TFIM: Allows for texel caching.
  - Reduction in memory traffic caused by forced re-computation.

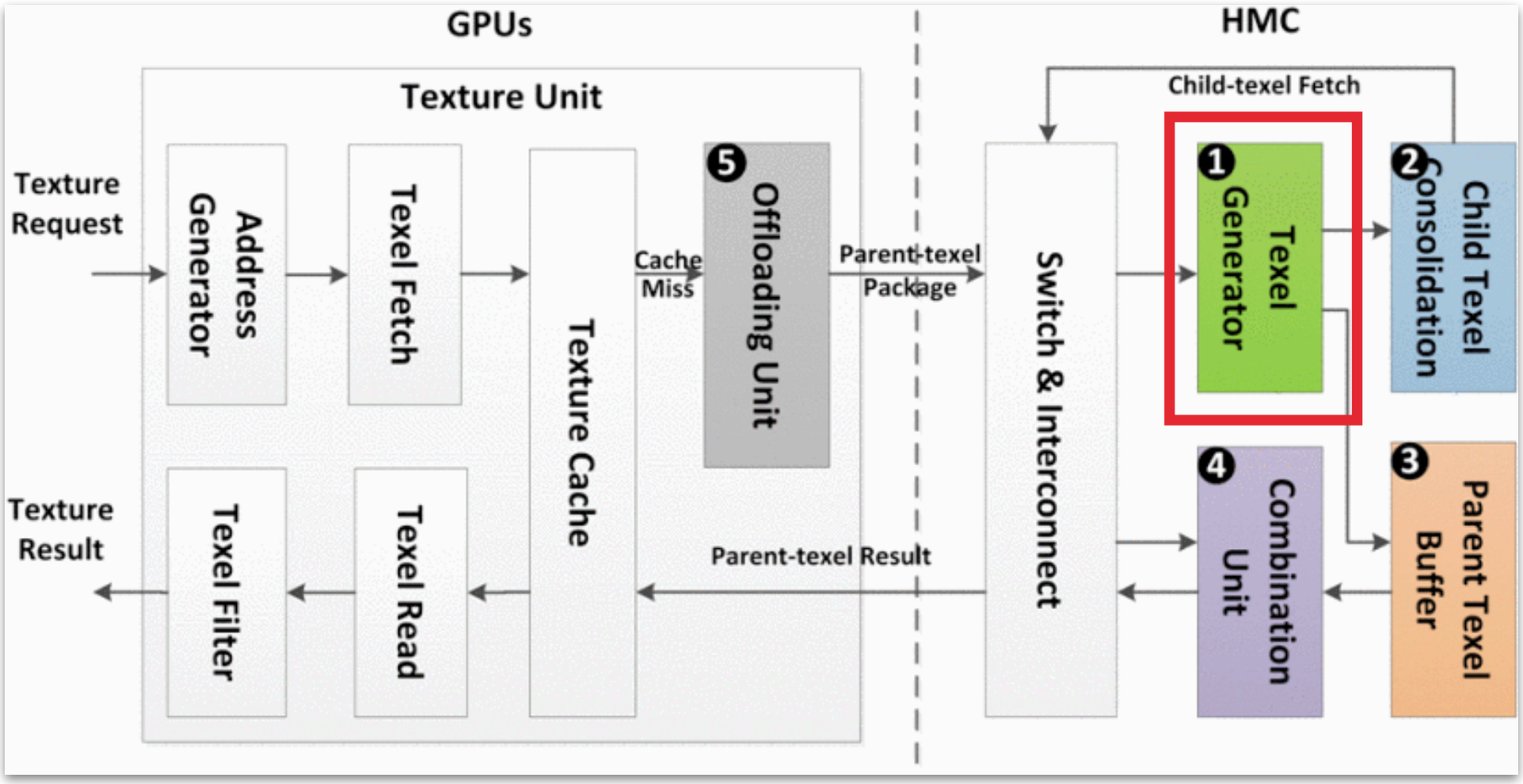
**Overview of A-TFIM Architecture**



# A-TFIM: ADVANCED TEXTURE FILTERING IN MEMORY

- A-TFIM: Allows for texel caching.
  - Reduction in memory traffic caused by forced re-computation.

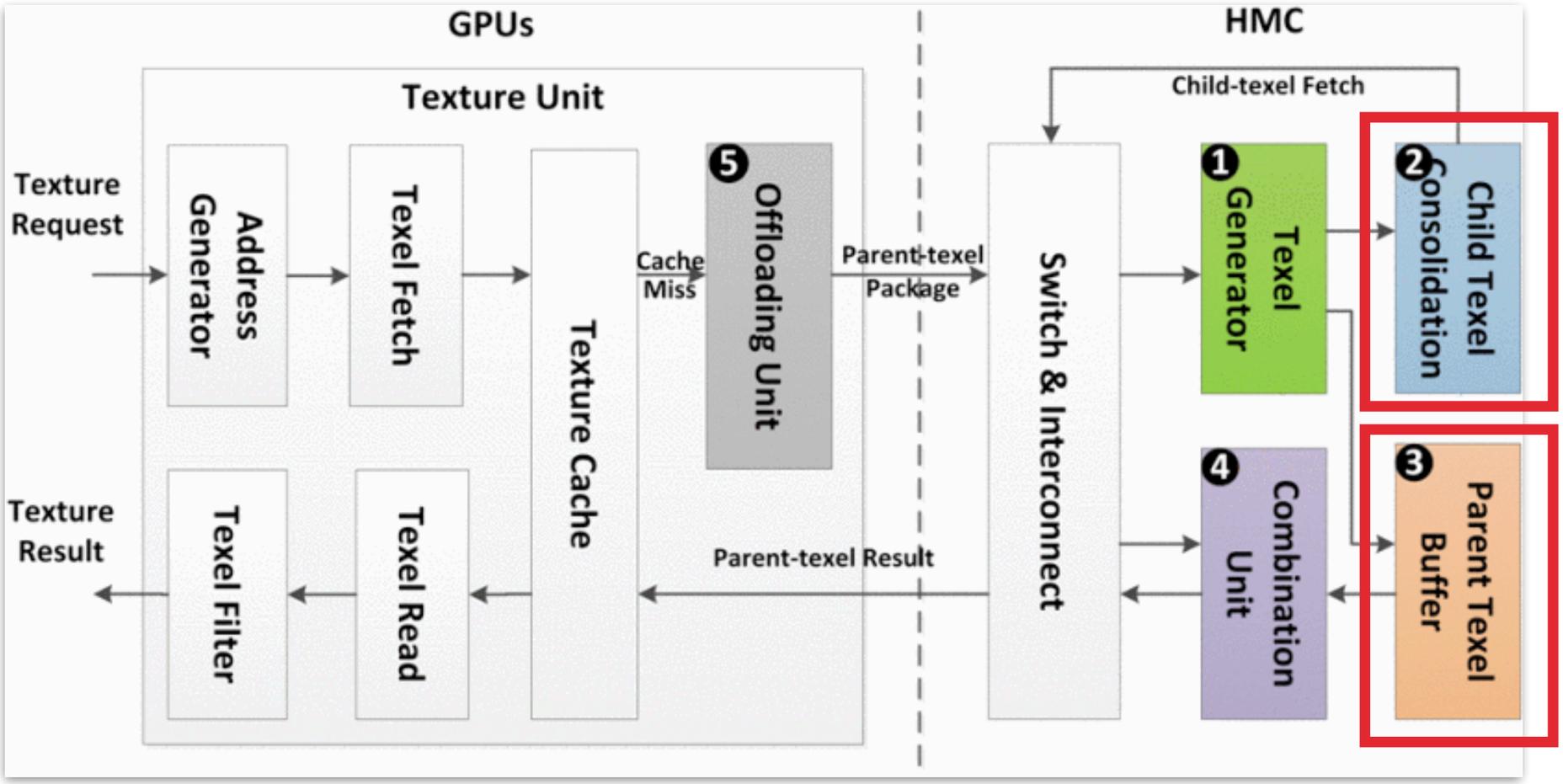
**Overview of A-TFIM Architecture**



# A-TFIM: ADVANCED TEXTURE FILTERING IN MEMORY

- A-TFIM: Allows for texel caching.
  - Reduction in memory traffic caused by forced re-computation.

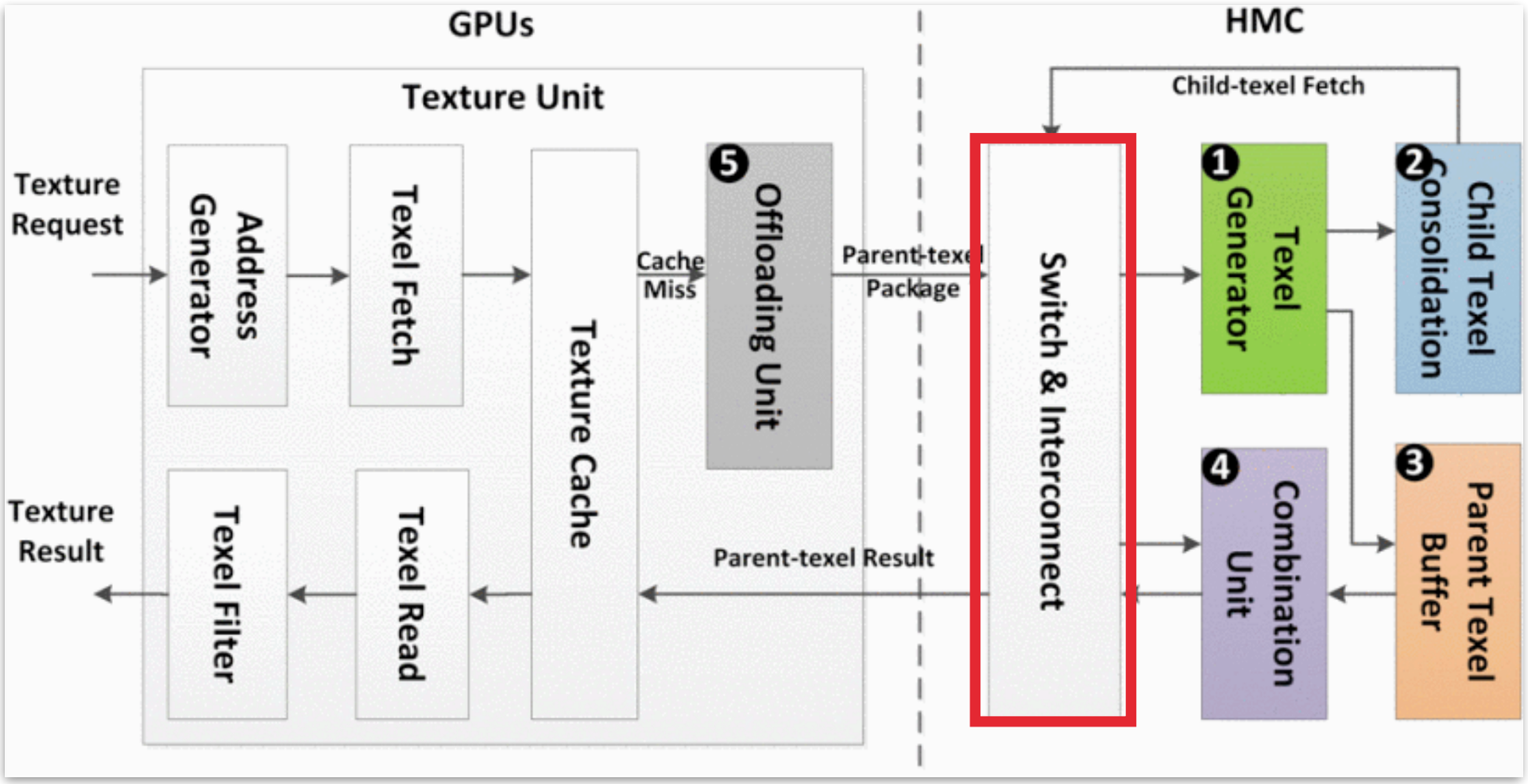
**Overview of A-TFIM Architecture**



# A-TFIM: ADVANCED TEXTURE FILTERING IN MEMORY

- A-TFIM: Allows for texel caching.
  - Reduction in memory traffic caused by forced re-computation.

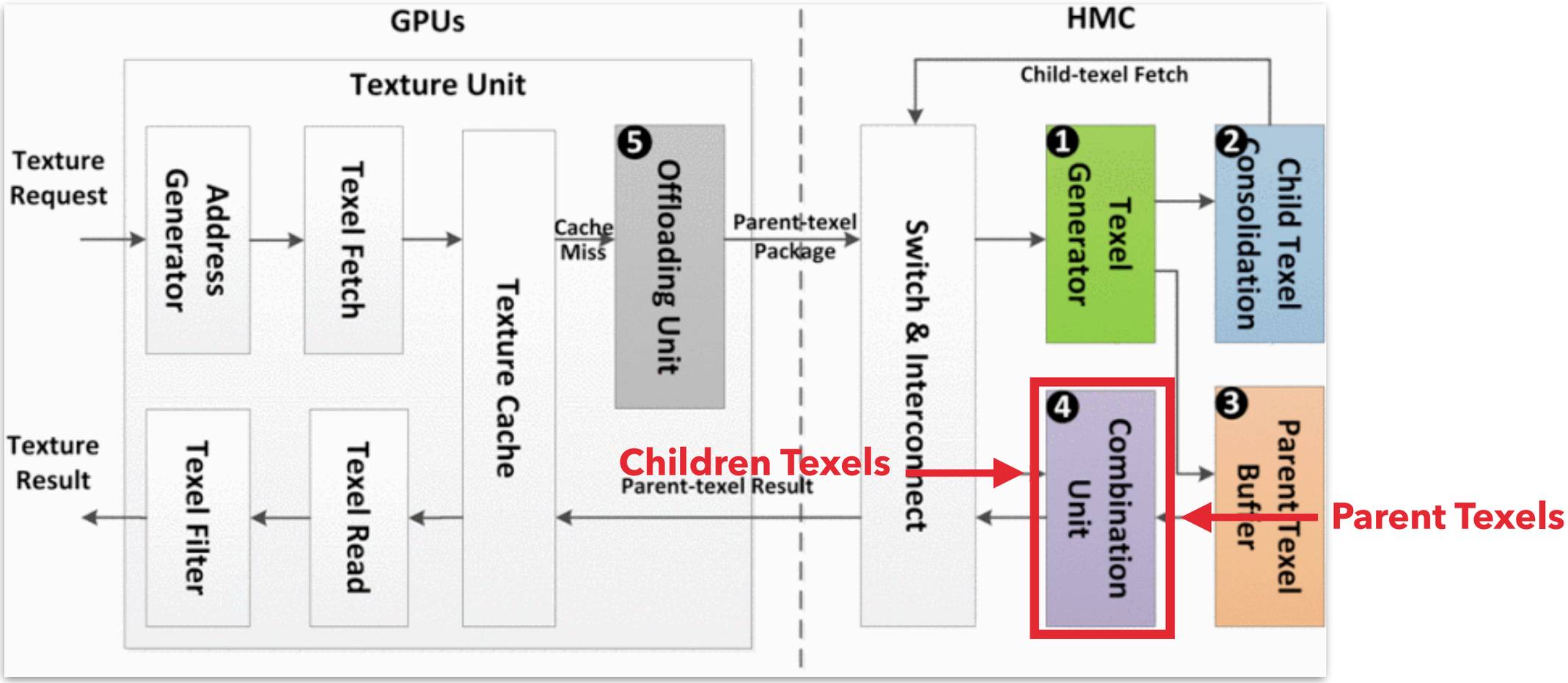
**Overview of A-TFIM Architecture**



# A-TFIM: ADVANCED TEXTURE FILTERING IN MEMORY

- A-TFIM: Allows for texel caching.
  - Reduction in memory traffic caused by forced re-computation.

**Overview of A-TFIM Architecture**

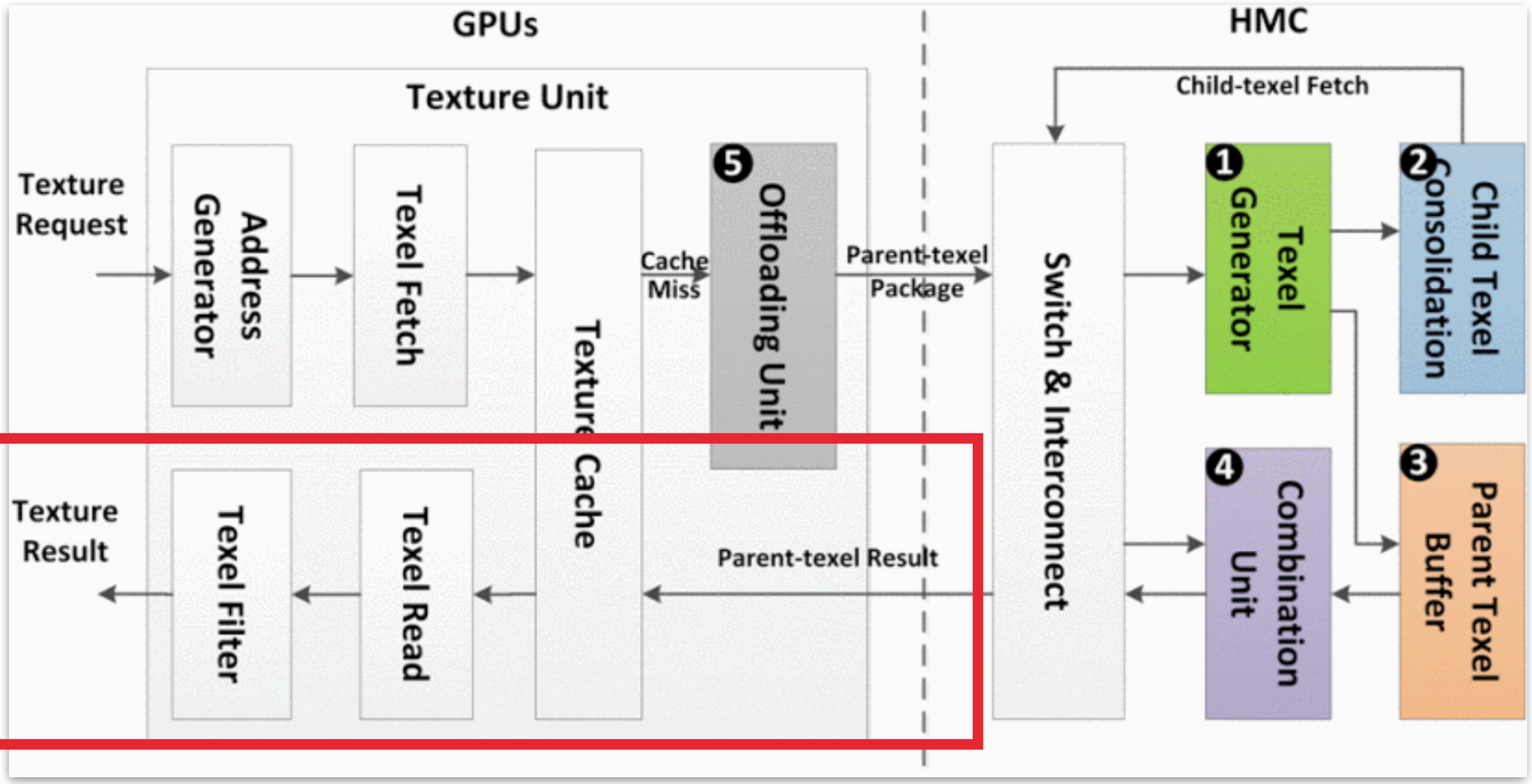



Actually do the anisotropic filtering

# A-TFIM: ADVANCED TEXTURE FILTERING IN MEMORY

- A-TFIM: Allows for texel caching.
  - Reduction in memory traffic caused by forced re-computation.

**Overview of A-TFIM Architecture**



# OUTLINE

- Background: Texture Filtering
- Motivation
- Solutions:
  - B-PIM: Basic Processing In Memory
  - S-TFIM: Simple Texture Filtering In Memory
  - A-TFIM: Advanced Texture Filtering In Memory
- Evaluation Methodology
- Evaluation Results
- Conclusion

# EVALUATION METHODOLOGY

# EVALUATION METHODOLOGY

- Simulator: **ATTILA** cycle accurate rasterization-based GPU sim.
  - Extended with an HMC block.

# EVALUATION METHODOLOGY

- Simulator: **ATTILA** cycle accurate rasterization-based GPU sim.
  - Extended with an HMC block.
- Power model: **McPAT** for power consumption of GPU's.

# EVALUATION METHODOLOGY

- Simulator: **ATTILA** cycle accurate rasterization-based GPU sim.
  - Extended with an HMC block.
- Power model: **McPAT** for power consumption of GPU's.
- How: Simulate extracted OpenGL and D3D commands.

# EVALUATION METHODOLOGY

- Simulator: **ATTILA** cycle accurate rasterization-based GPU sim.
  - Extended with an HMC block.
- Power model: **McPAT** for power consumption of GPU's.
- How: Simulate extracted OpenGL and D3D commands.
- Use Cases: 3D video-games: doom3, Fear, and Half-Life 2.

# EVALUATION METHODOLOGY

- Simulator: **ATTILA** cycle accurate rasterization-based GPU sim.
  - Extended with an HMC block.
- Power model: **McPAT** for power consumption of GPU's.
- How: Simulate extracted OpenGL and D3D commands.
- Use Cases: 3D video-games: doom3, Fear, and Half-Life 2.
- Metrics:

# EVALUATION METHODOLOGY

- Simulator: **ATTILA** cycle accurate rasterization-based GPU sim.
  - Extended with an HMC block.
- Power model: **McPAT** for power consumption of GPU's.
- How: Simulate extracted OpenGL and D3D commands.
- Use Cases: 3D video-games: doom3, Fear, and Half-Life 2.
- Metrics:
  - **Quality** in **Peak Signal-to-Noise Ratio** (PSNR).

# EVALUATION METHODOLOGY

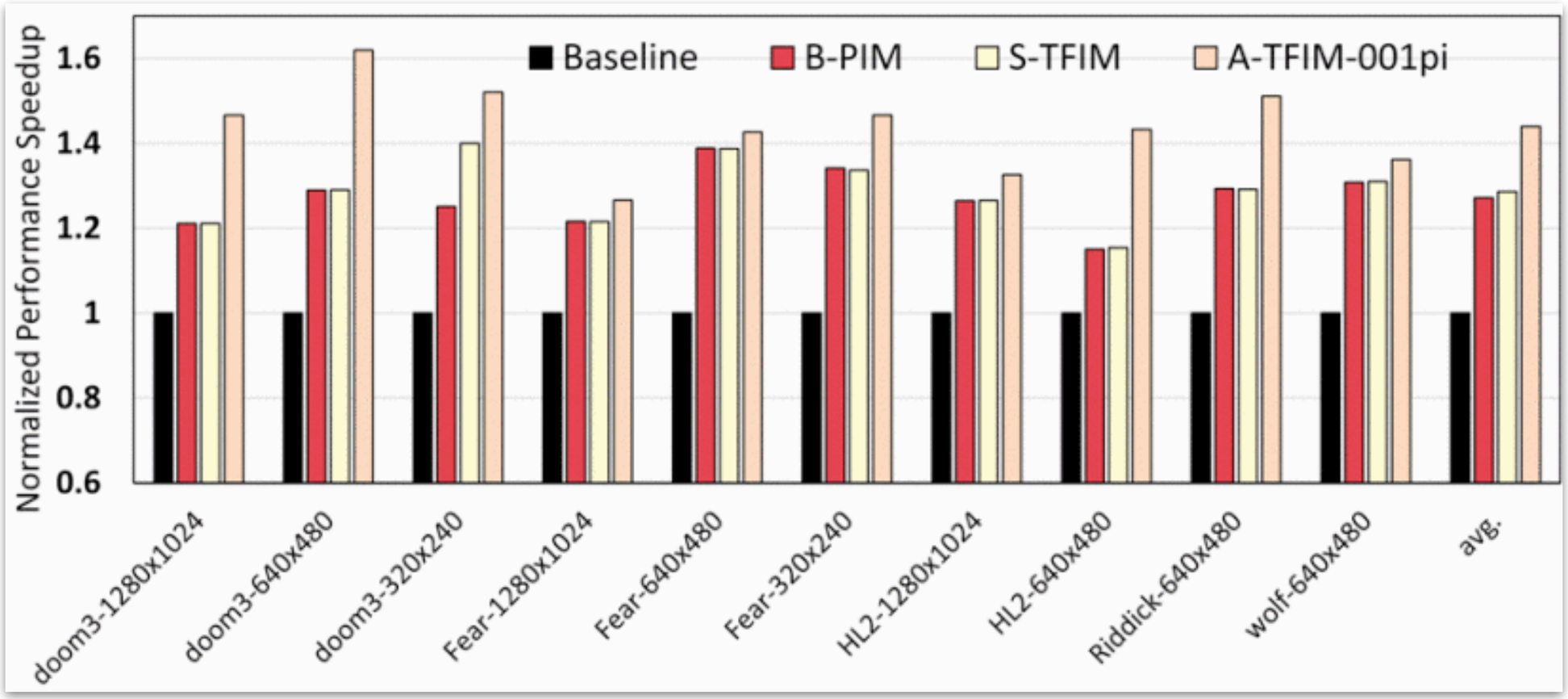
- Simulator: **ATTILA** cycle accurate rasterization-based GPU sim.
  - Extended with an HMC block.
- Power model: **McPAT** for power consumption of GPU's.
- How: Simulate extracted OpenGL and D3D commands.
- Use Cases: 3D video-games: doom3, Fear, and Half-Life 2.
- Metrics:
  - **Quality** in **Peak Signal-to-Noise Ratio** (PSNR).
  - **Performance** in **Normalized Speedup.**

# OUTLINE

- Background: Texture Filtering
- Motivation
- Solutions:
  - B-PIM: Basic Processing In Memory
  - S-TFIM: Simple Texture Filtering In Memory
  - A-TFIM: Advanced Texture Filtering In Memory
- Evaluation Methodology
- Evaluation Results
- Conclusion

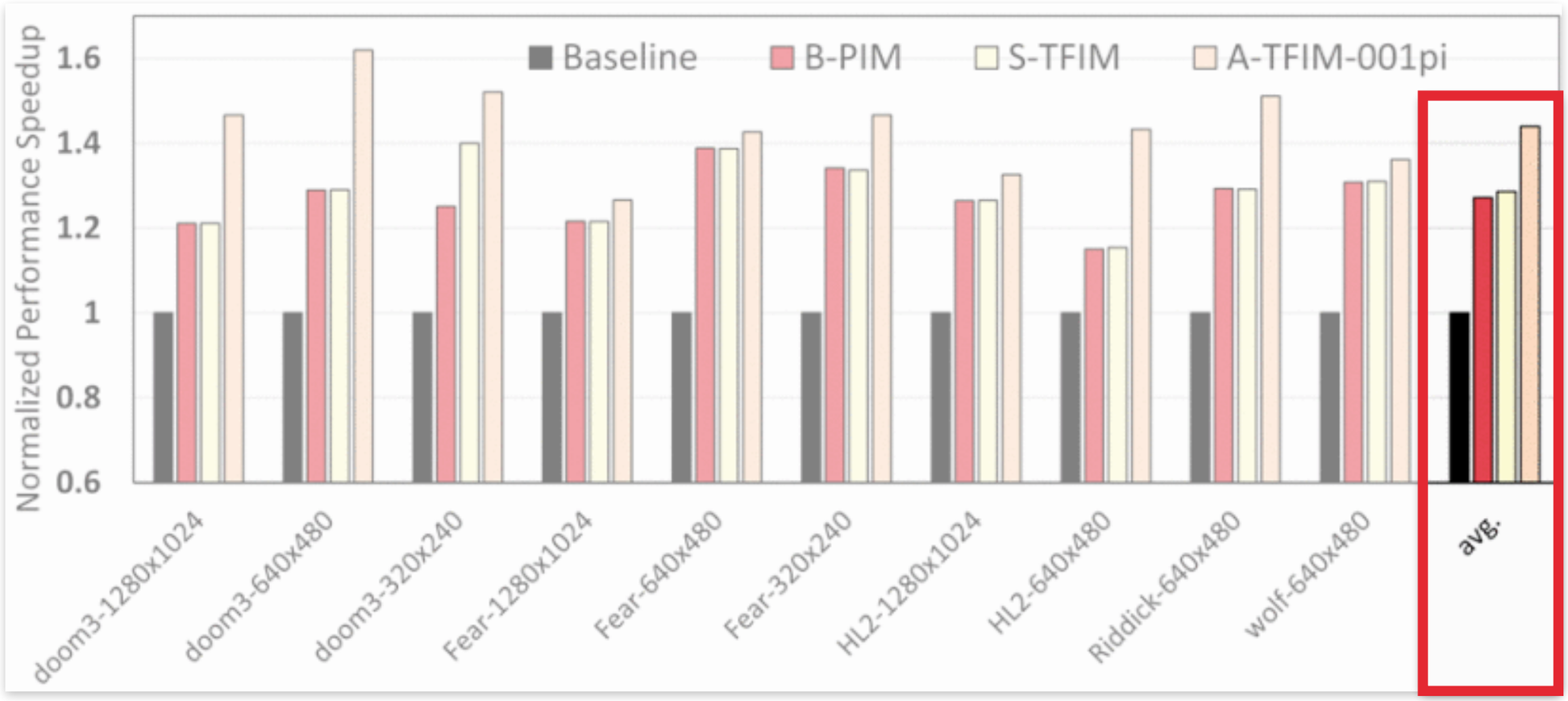
# RESULTS: 3D RENDERING PERFORMANCE

**Results from Overall 3D Rendering Performance Evaluation**



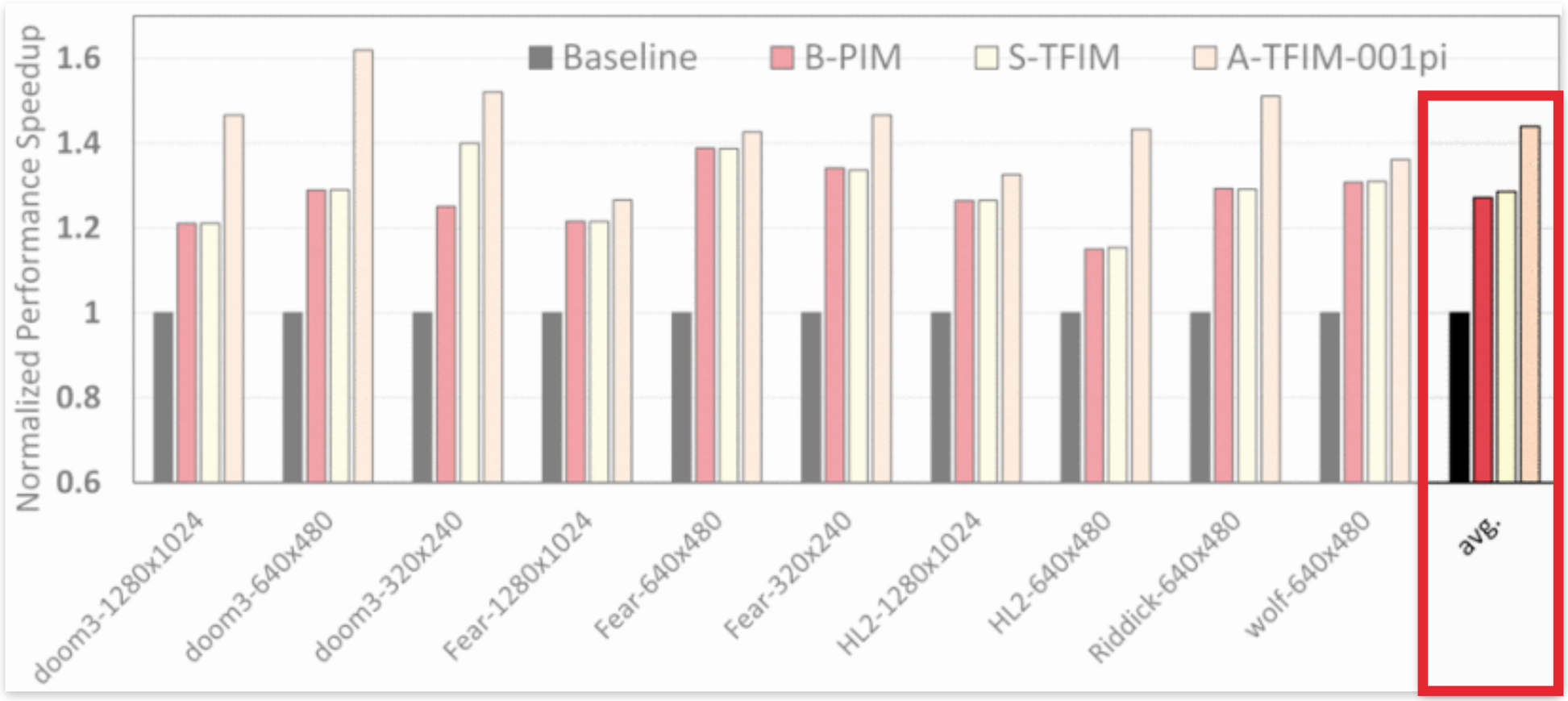
# RESULTS: 3D RENDERING PERFORMANCE

**Results from Overall 3D Rendering Performance Evaluation**



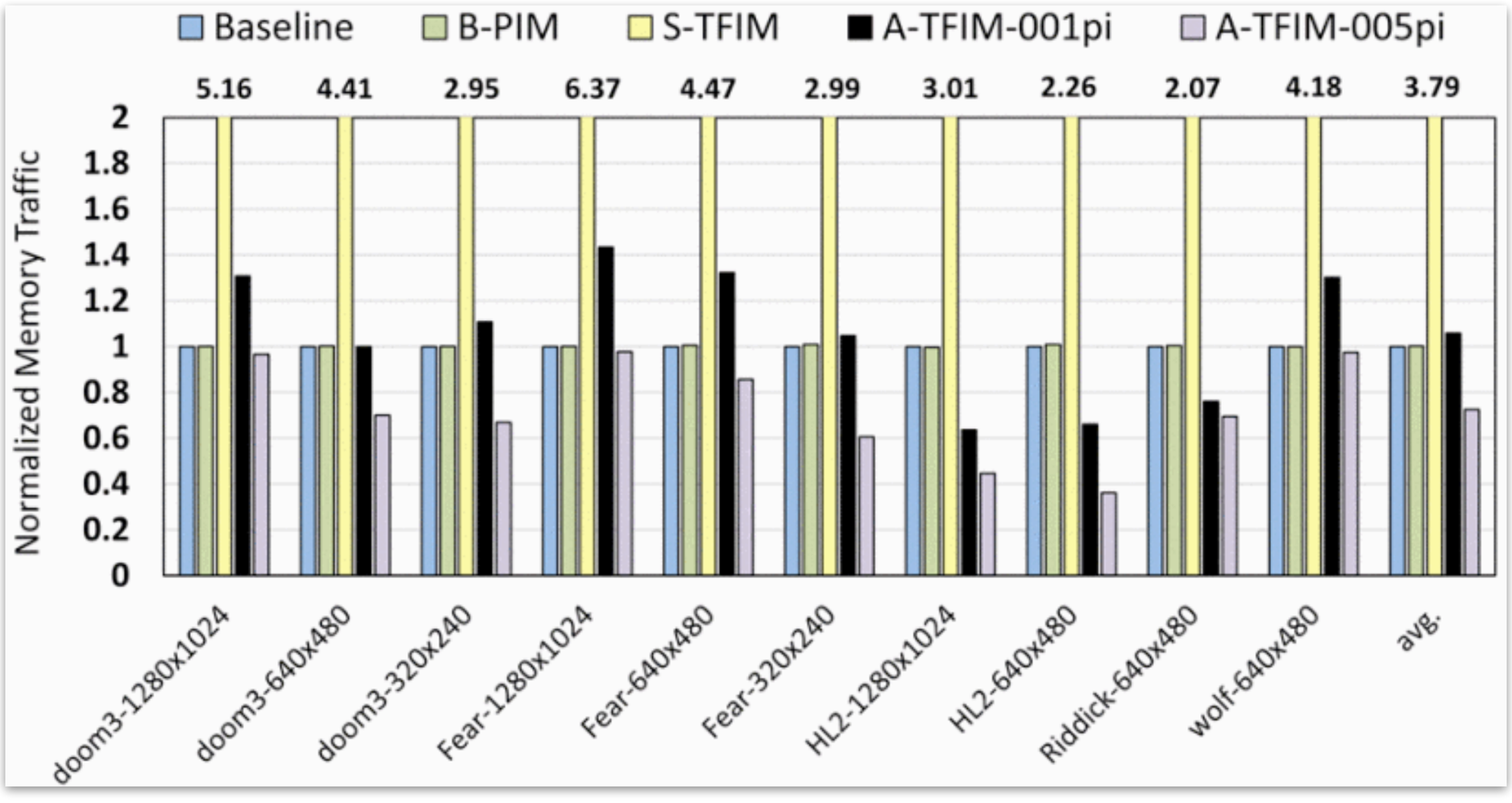
# RESULTS: 3D RENDERING PERFORMANCE

**Results from Overall 3D Rendering Performance Evaluation**

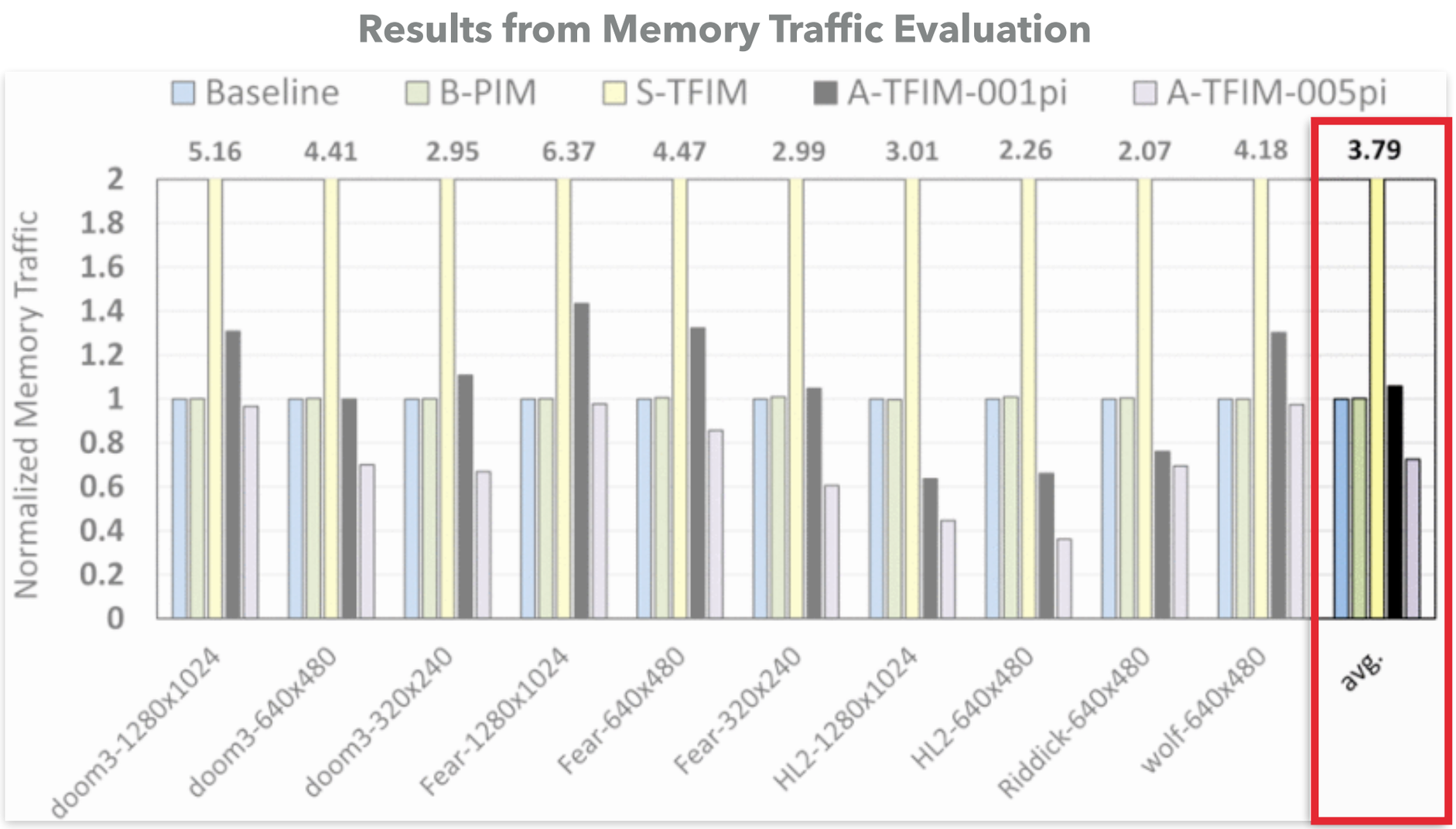



- S-TFIM: Same performance as B-PIM.
- A-TFIM: **1.5x speedup** over baseline GPU.

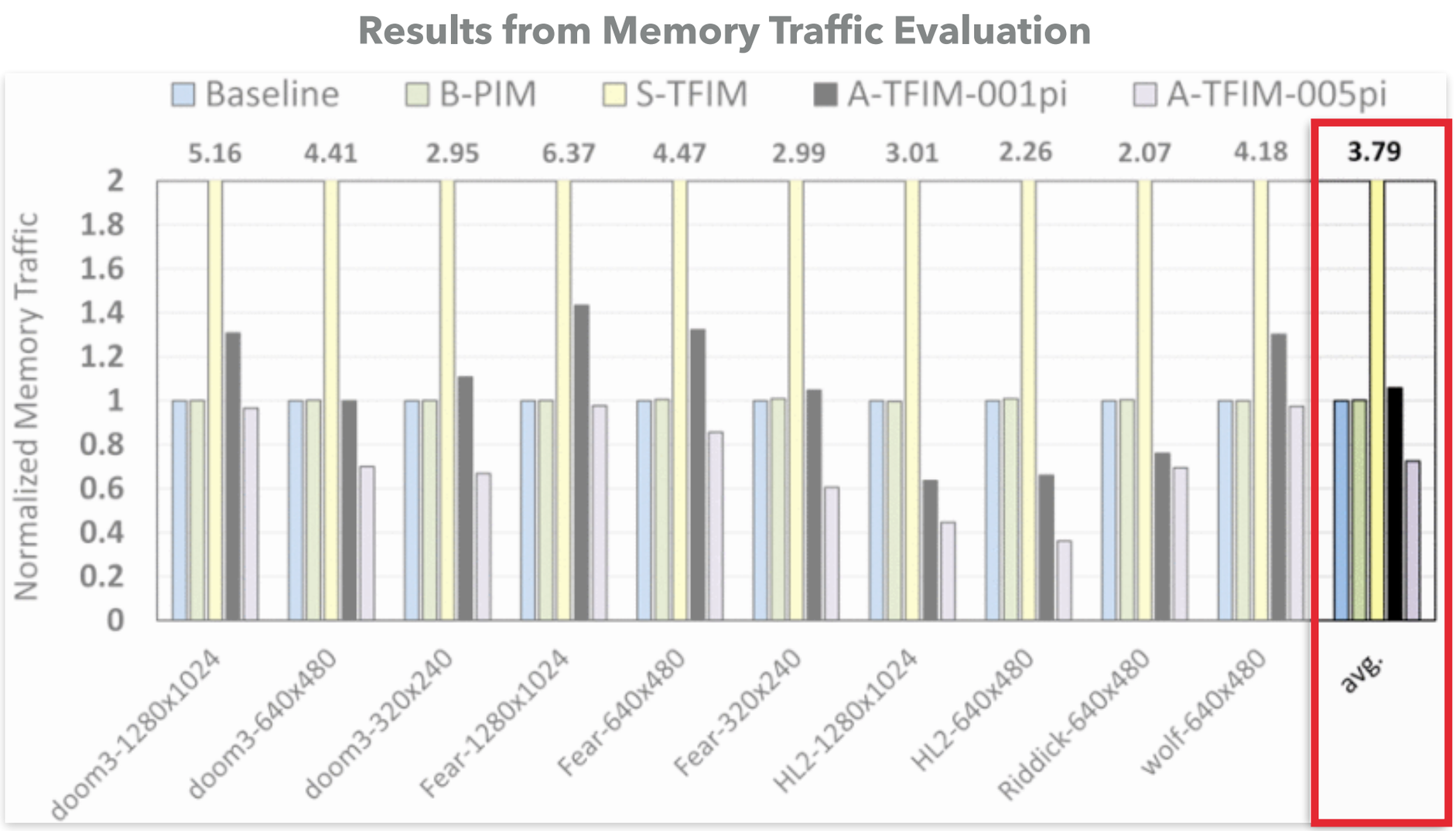
# RESULTS: MEMORY TRAFFIC



# RESULTS: MEMORY TRAFFIC

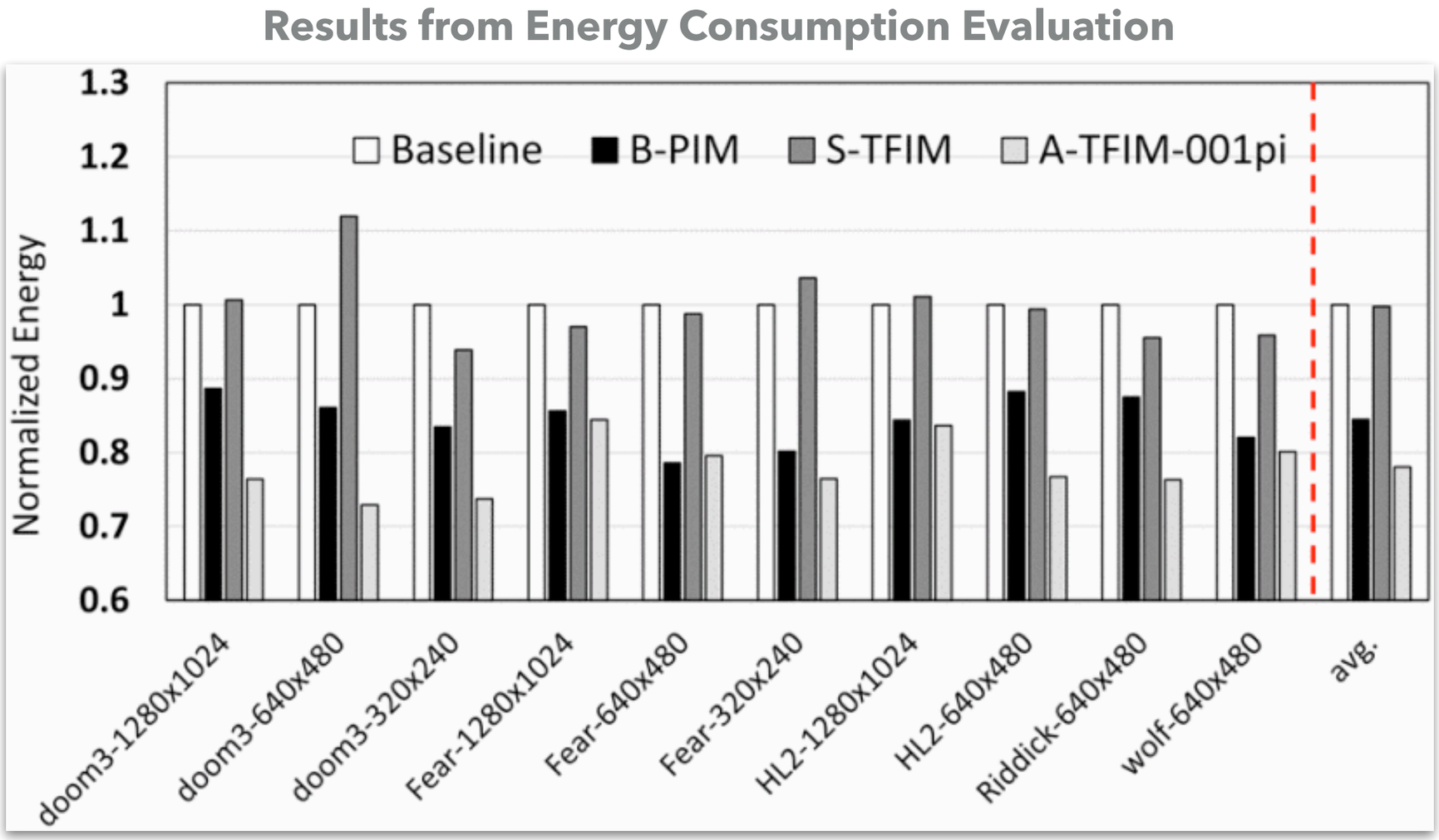


# RESULTS: MEMORY TRAFFIC

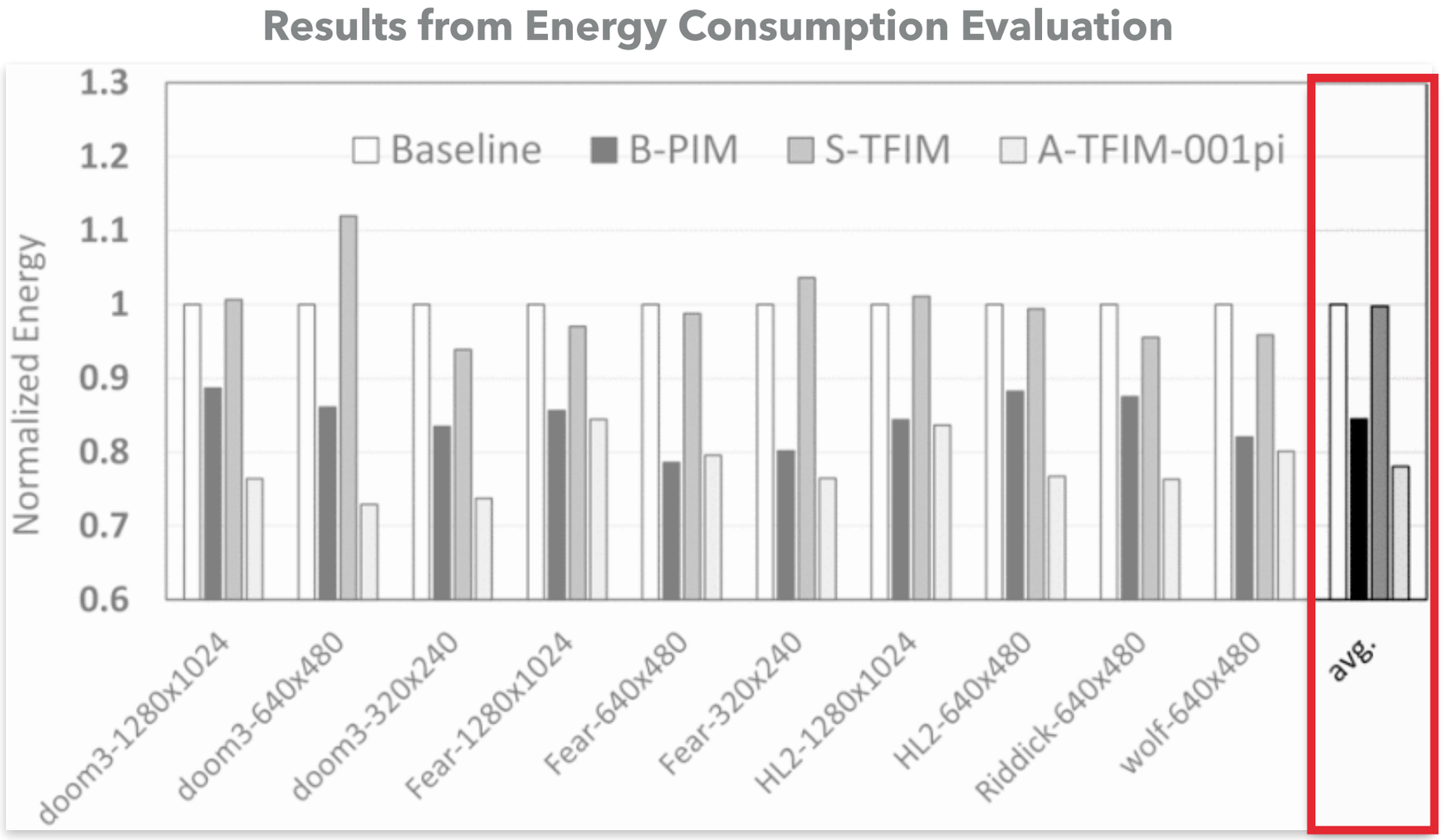



- S-TFIM: Forced re-computation of intermediate texels due to lack of texel-caching in GPU ==> **High memory traffic**.

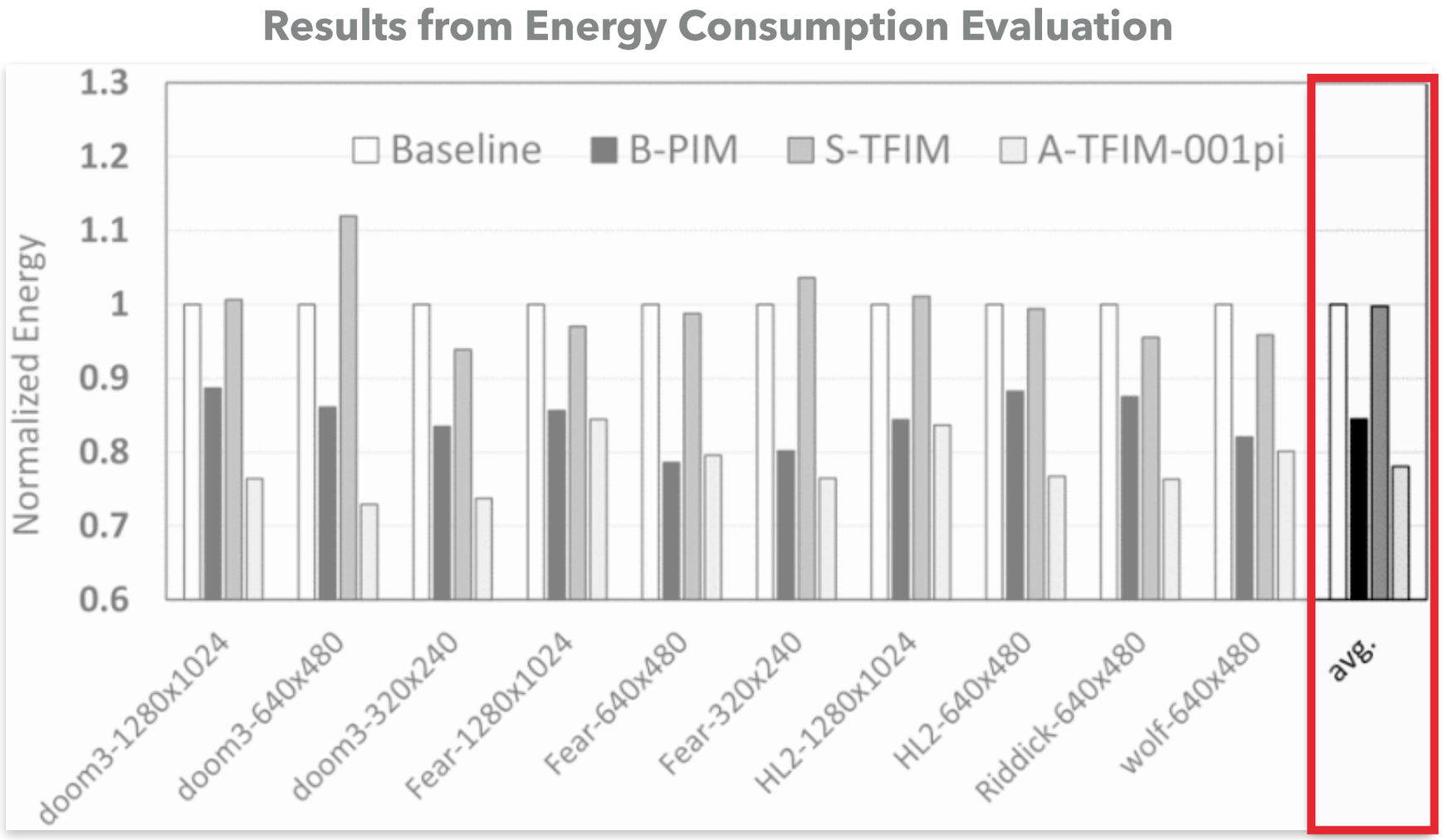
# RESULTS: ENERGY CONSUMPTION



# RESULTS: ENERGY CONSUMPTION



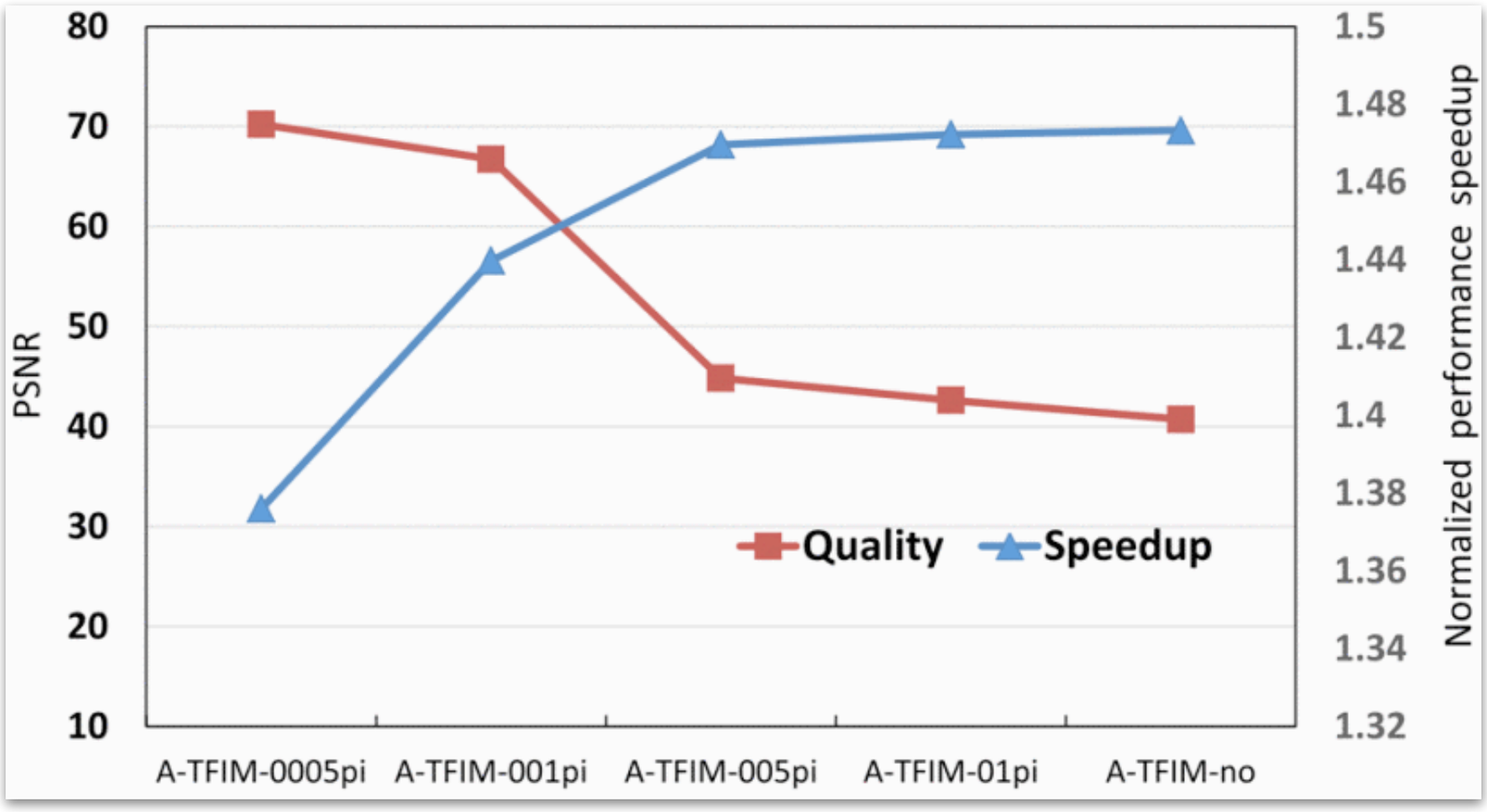
# RESULTS: ENERGY CONSUMPTION




- A-TFIM: Reduces energy consumption by **25%** over baseline GPU.

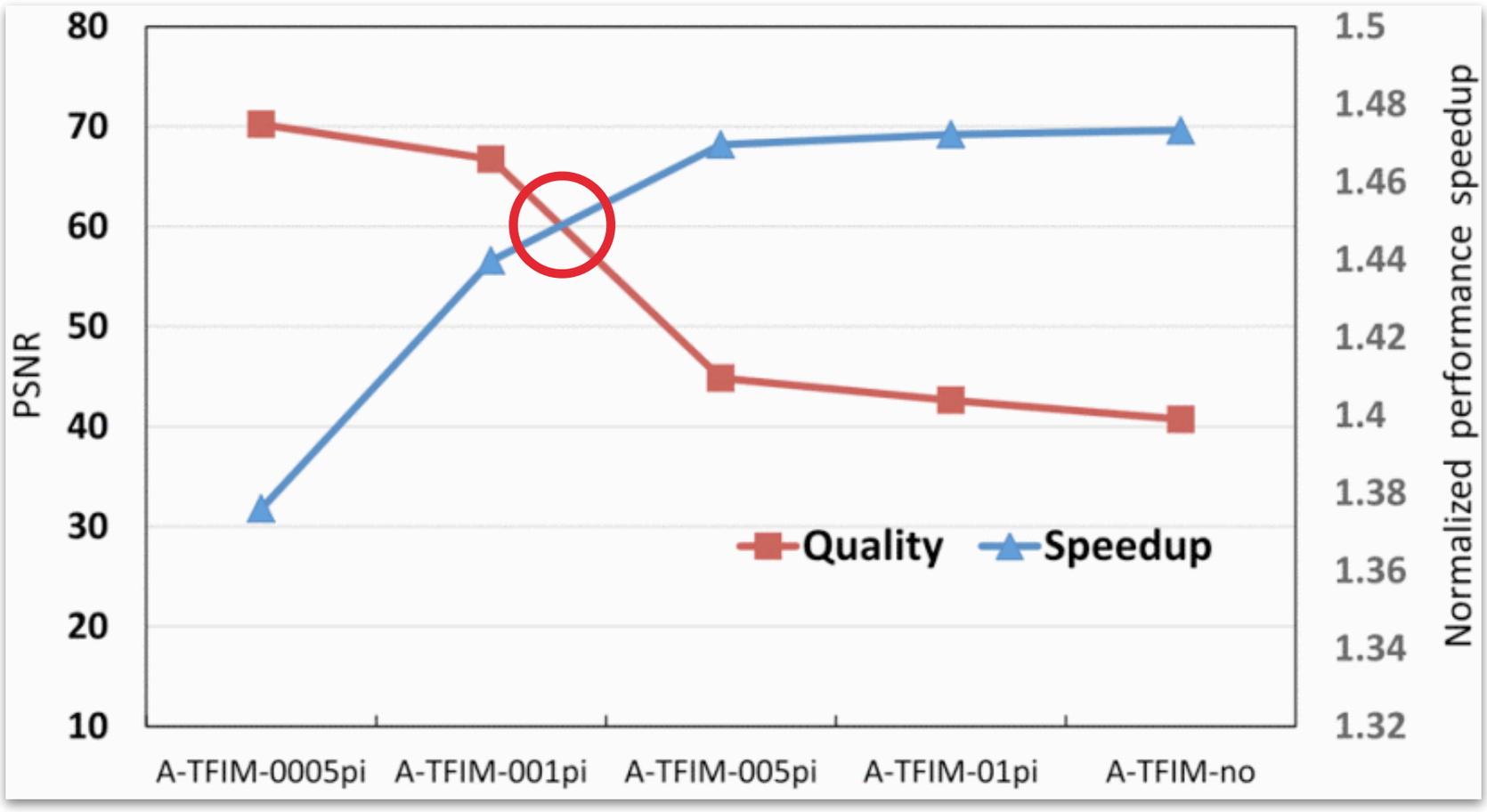
# RESULTS: QUALITY VS. SPEEDUP FOR A-TFIM ANGLES

**Results from Camera Angle Threshold Quality reduction Evaluation**



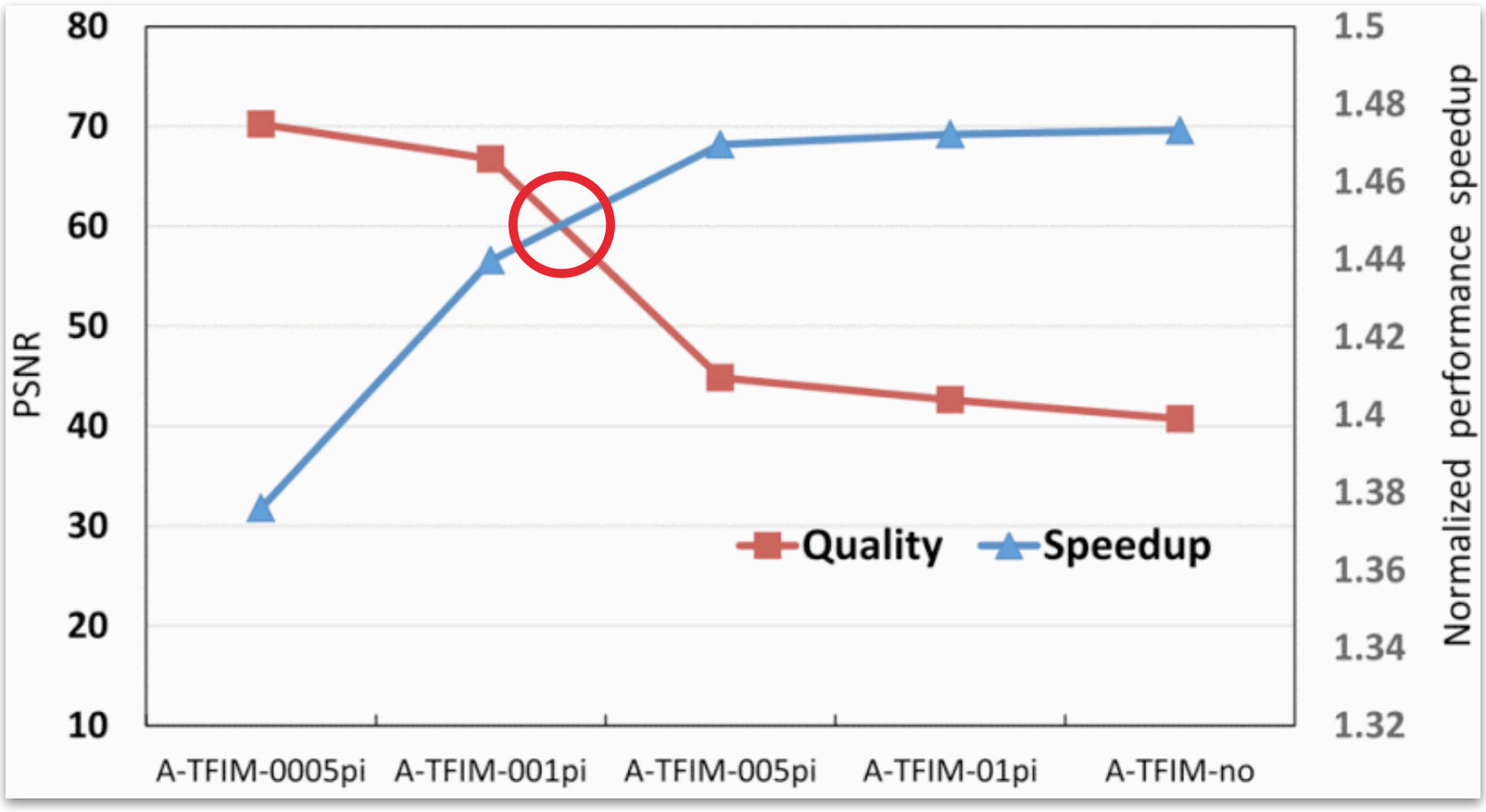
# RESULTS: QUALITY VS. SPEEDUP FOR A-TFIM ANGLES

**Results from Camera Angle Threshold Quality reduction Evaluation**



# RESULTS: QUALITY VS. SPEEDUP FOR A-TFIM ANGLES

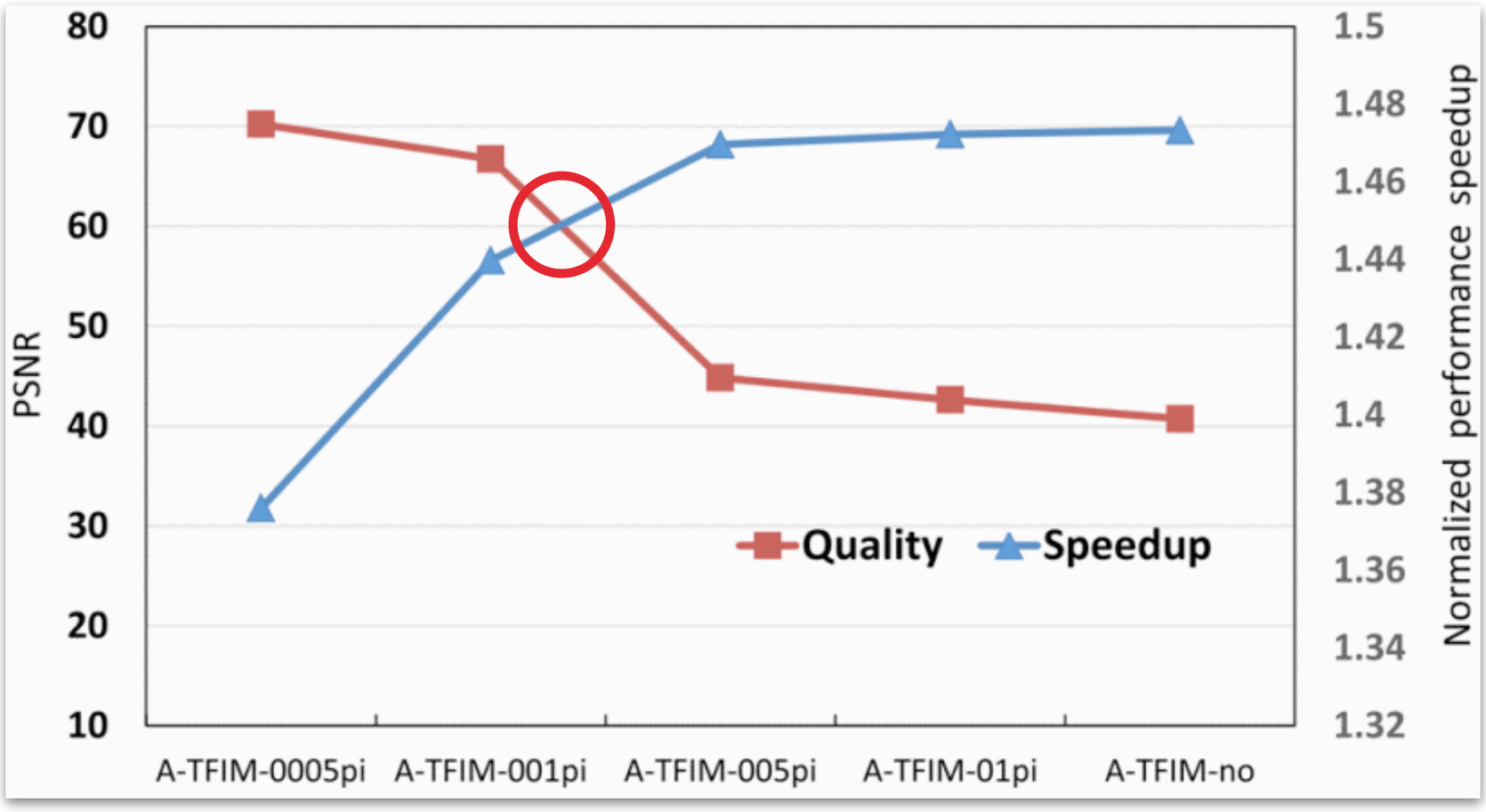
**Results from Camera Angle Threshold Quality reduction Evaluation**




- **Lowest threshold** yields **virtually perfect image** but **lower speedup**.

# RESULTS: QUALITY VS. SPEEDUP FOR A-TFIM ANGLES

**Results from Camera Angle Threshold Quality reduction Evaluation**




- **Lowest threshold** yields **virtually perfect image** but **lower speedup**.
- Ideal threshold: 0.01pi ≈ 1.8°

# OUTLINE

- Background: Texture Filtering
- Motivation
- Solutions:
  - B-PIM: Basic Processing In Memory
  - S-TFIM: Simple Texture Filtering In Memory
  - A-TFIM: Advanced Texture Filtering In Memory
- Evaluation Methodology
- Evaluation Results
- Conclusion

# CONCLUSION

- **Motivation**: Texture filtering = **60%** of **memory requests** in real-time rendering applications.
- **Problem**: Real-time rendering is a highly memory bound task due to texture filtering.
- **Idea**: Bypass memory bottleneck caused by texture filtering by:
  - Using **more** *memory efficient* filtering pipeline.
  - **Reducing** data movement in system.
- **Solution**: Advanced-Texture Filtering In Memory (A-TFIM).
  - Implements a portion of the **texture filtering pipeline** in **HMC**.
  - Uses a **camera-angle threshold** for **performance/accuracy** ratio control.
- **Key results**: Evaluated on various 3D video-games.
  - Average **1.4x** rendering speedup over baseline.
  - Average **22% less** energy consumption compared to baseline.

# QUESTIONS?

# REVIEW

# STRENGTHS

# STRENGTHS

- Proposed solution is **novel**, very little work exists in the field of PIM-enabled Graphics.

# STRENGTHS

- Proposed solution is **novel**, very little work exists in the field of PIM-enabled Graphics.
- Results obtained are impressive in terms of energy efficiency, which would be **great for mobile rendering**.

# STRENGTHS

- Proposed solution is **novel**, very little work exists in the field of PIM-enabled Graphics.
- Results obtained are impressive in terms of energy efficiency, which would be **great for mobile rendering**.
- The proposed pipeline reordering with low precision loss requires **domain-expertise** from the authors to figure out, showing the authors' investment in the field.

# WEAKNESSES

# WEAKNESSES

- Reminder: The goal of this project is to improve 3D video-game performance.

# WEAKNESSES

- Reminder: The goal of this project is to improve 3D video-game performance.
- Many different works have tackled this, could lead to **better results** with **simpler implementations** (look at texture compression for example).

# WEAKNESSES

- Reminder: The goal of this project is to improve 3D video-game performance.
- Many different works have tackled this, could lead to **better results** with **simpler implementations** (look at texture compression for example).
- The system was **only tested on games from 20+years ago**, rendering demands have greatly changed since then, how does this perform on modern applications?

# TAKEAWAY

# TAKEAWAY

- Accelerating Graphics is a problem that is difficult to approach since it requires domain-specific expertise in both Computer Graphics and Computer Architecture.

# TAKEAWAY

- Accelerating Graphics is a problem that is difficult to approach since it requires domain-specific expertise in both Computer Graphics and Computer Architecture.
- This work proves that PIM architecture can be used to construct better accelerators for 3D rendering.

# TAKEAWAY

- Accelerating Graphics is a problem that is difficult to approach since it requires domain-specific expertise in both Computer Graphics and Computer Architecture.
- This work proves that PIM architecture can be used to construct better accelerators for 3D rendering.
- Currently feels like a “proof of concept” and needs to be adapted for modern workloads.

# DISCUSSION: CAN GRAPHICS BENEFIT FROM PIM?

# DISCUSSION: CAN GRAPHICS BENEFIT FROM PIM?

- Modern systems have been evolving and taking many different forms.

# DISCUSSION: CAN GRAPHICS BENEFIT FROM PIM?

- Modern systems have been evolving and taking many different forms.
- Rendering is now done on all types of systems: embedded VR, Cloud gaming, mobile rendering, …

# DISCUSSION: CAN GRAPHICS BENEFIT FROM PIM?

- Modern systems have been evolving and taking many different forms.
- Rendering is now done on all types of systems: embedded VR, Cloud gaming, mobile rendering, …
- Follow-up work: *"PIM-VR: Erasing Motion Anomalies In Highly-Interactive Virtual Reality World with Customized Memory Cube"*

# DISCUSSION: CAN GRAPHICS BENEFIT FROM PIM?

- Modern systems have been evolving and taking many different forms.
- Rendering is now done on all types of systems: embedded VR, Cloud gaming, mobile rendering, …
- Follow-up work: *"PIM-VR: Erasing Motion Anomalies In Highly-Interactive Virtual Reality World with Customized Memory Cube"*
- **Question: Can these new systems benefit from using PIM architectures?**

# DISCUSSION: PIM ARCHITECTURES

# DISCUSSION: PIM ARCHITECTURES

- Most applications have their own memory and compute bound parts.

# DISCUSSION: PIM ARCHITECTURES

- Most applications have their own memory and compute bound parts.
- Most of the work in PIM architectures is in isolating these parts from the entire application.

# DISCUSSION: PIM ARCHITECTURES

- Most applications have their own memory and compute bound parts.
- Most of the work in PIM architectures is in isolating these parts from the entire application.
- **Question: How to decide when/what to offload to PIM systems? Can this be automated? Is a cache miss always the best decision metric?**

# DISCUSSION: IS THERE A BETTER APPROACH?

# DISCUSSION: IS THERE A BETTER APPROACH?

- Texture filtering can be accelerated with other methods such as smart prefetching or texture compression.

# DISCUSSION: IS THERE A BETTER APPROACH?

- Texture filtering can be accelerated with other methods such as smart prefetching or texture compression.
- Modern rendering applications struggle more with **unpredictable memory latencies** from geometry fetches rather than textures.

# DISCUSSION: IS THERE A BETTER APPROACH?

- Texture filtering can be accelerated with other methods such as smart prefetching or texture compression.
- Modern rendering applications struggle more with **unpredictable memory latencies** from geometry fetches rather than textures.
- **<u>Question</u>: Can we leverage PIM architectures for handling unpredictable memory latencies?**

# ACKNOWLEDGEMENTS


- Big thanks to my mentors:
  - Geraldo De Oliveira
  - Mohammad Sadrosadati
  - Haocong Luo
- For all of their feedback throughout the (many) iterations of this presentation.

# LET'S DISCUSS

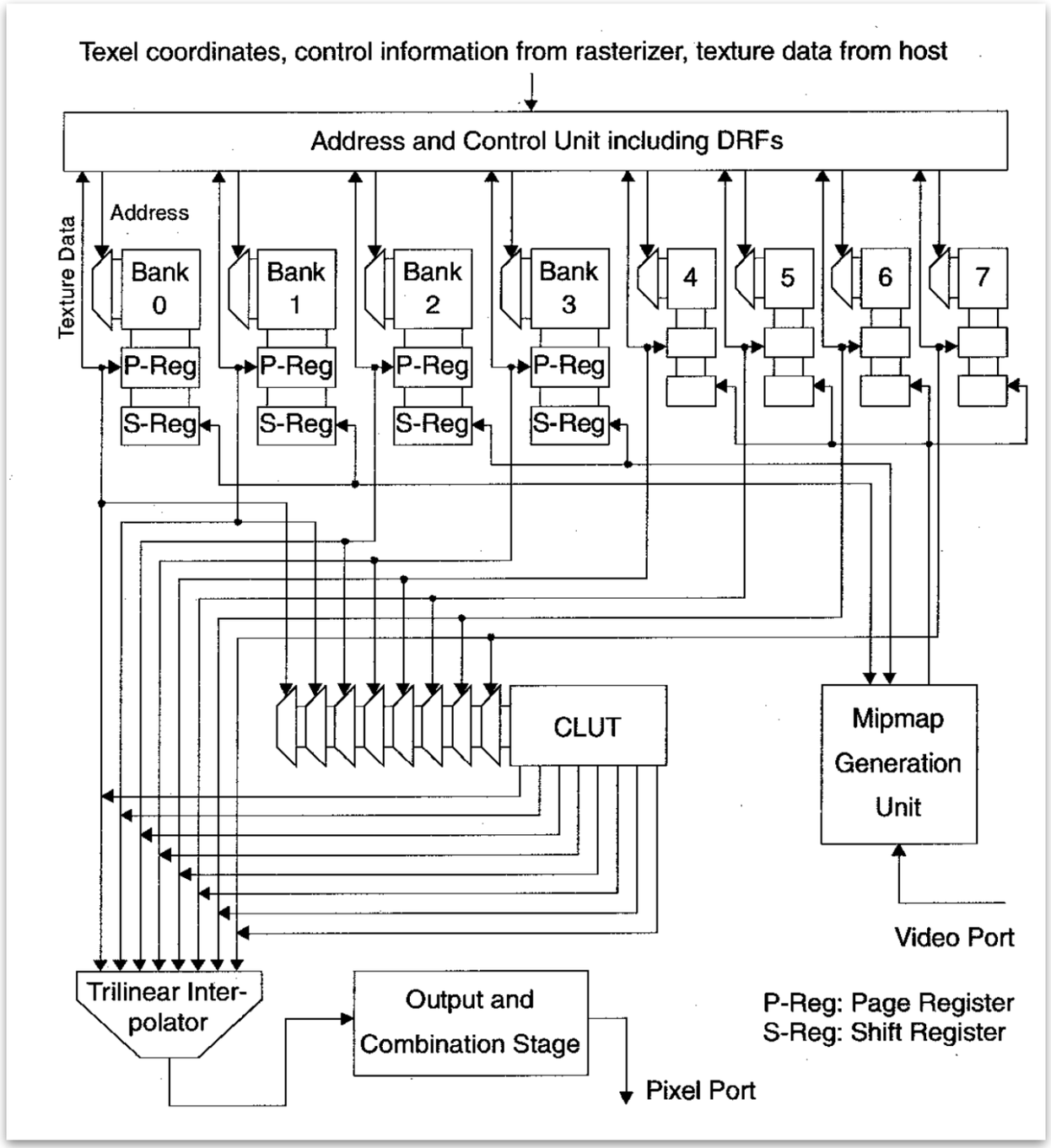
# PRIOR WORK: TEXRAM, A SMART MEMORY FOR TEXTURING

**Texram chip:**

Separate chip with its own memory capable of generating Mipmaps and performing Bilinear and Trilinear filtering.

**Reference:**

*Andreas Schilling, Gunter Knittel, and Wolfgang Strasser - IEEE Computer Graphics and Applications 1996*



# FOLLOW-UP WORK: PIM-VR

- **Idea**: Reduce head movement response delay in Virtual Reality systems by accelerating Asynchronous Time Warp (ATW) using a PIM architecture.
- **Reference**: (Same authors as this work) *"PIM-VR: Erasing Motion Anomalies In Highly-Interactive Virtual Reality World With Customized Memory Cube" - HPCA 2019*