



**SIGGRAPH2006**



SIGGRAPH2006

---

# Fast Median and Bilateral Filtering

Ben Weiss, Shell & Slate Software



SIGGRAPH2006

# Why Median?

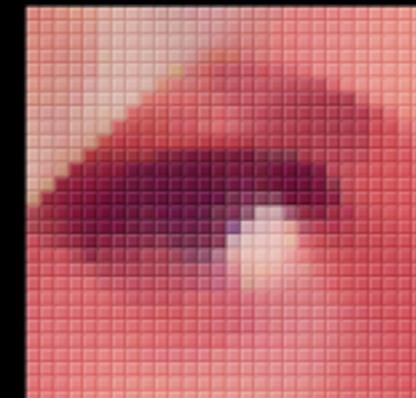
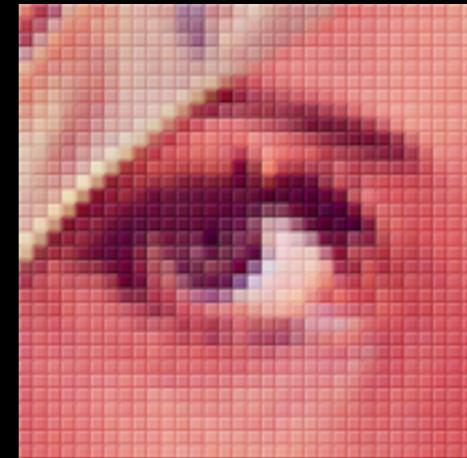
---



SIGGRAPH2006

# Why Median?

- Fundamental

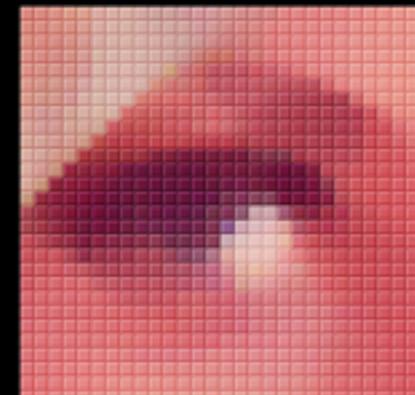
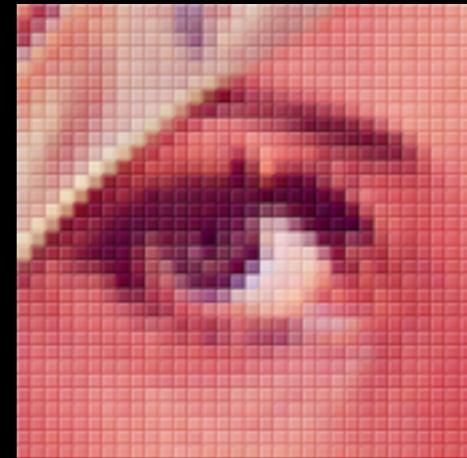




SIGGRAPH2006

# Why Median?

- Fundamental
  - Mean Minimizes  $|\sum I_p - m|$



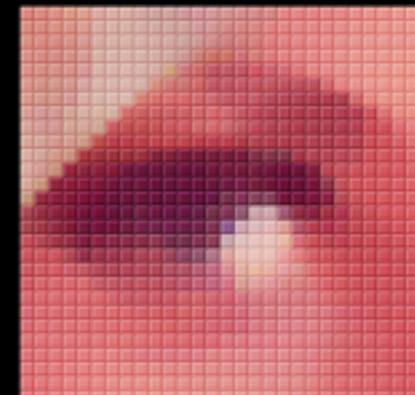
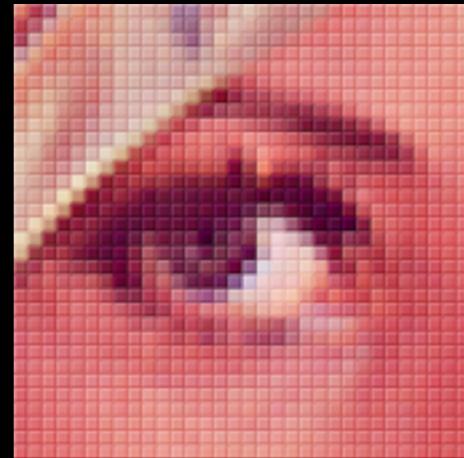


SIGGRAPH2006

# Why Median?

- Fundamental

- Mean Minimizes  $|\sum I_p - m|$
- Median Minimizes  $\sum |I_p - m|$





SIGGRAPH2006

# Why Median?

- Fundamental
  - Mean Minimizes  $\left| \sum I_p - m \right|$
  - Median Minimizes  $\sum |I_p - m|$



Gaussian



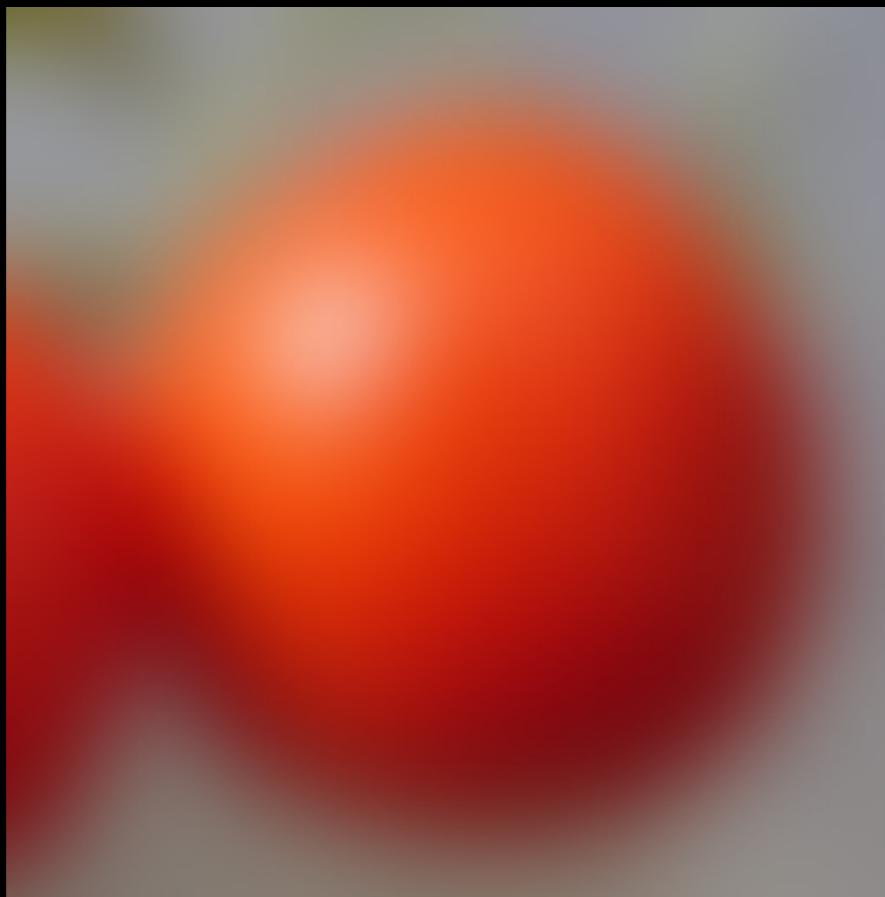
Median



SIGGRAPH2006

# Why Median?

- Fundamental
  - Mean Minimizes  $|\sum I_p - m|$
  - Median Minimizes  $\sum |I_p - m|$



Gaussian



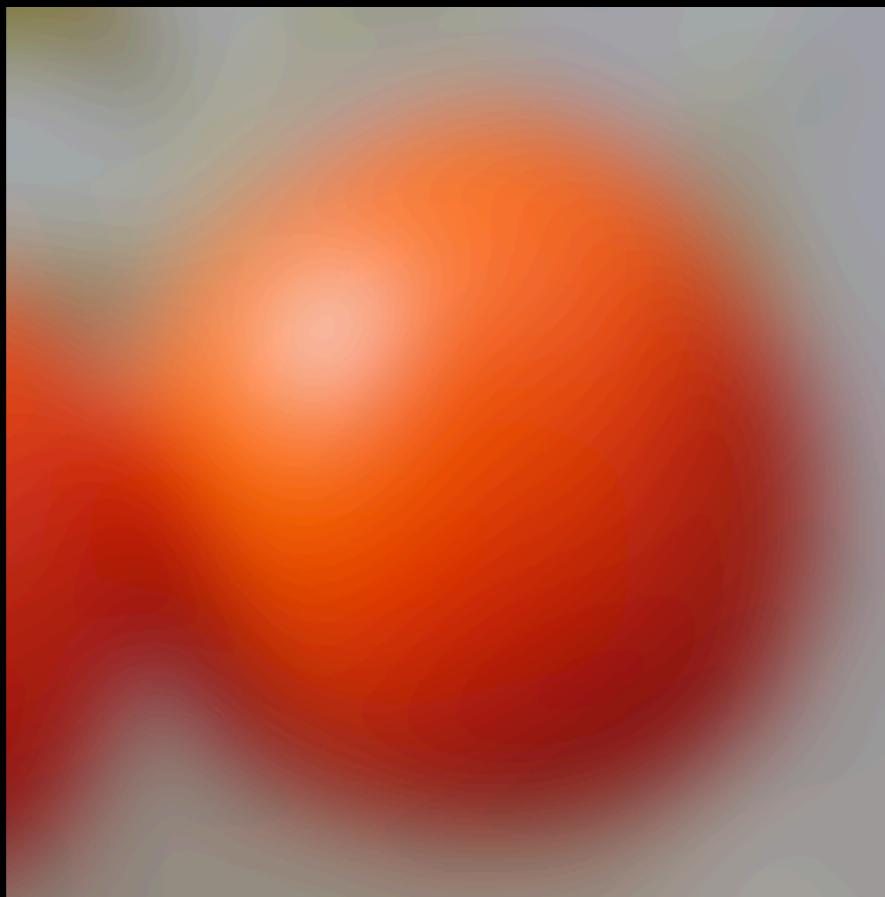
Median



SIGGRAPH2006

# Why Median?

- Fundamental
  - Mean Minimizes  $\left| \sum I_p - m \right|$
  - Median Minimizes  $\sum |I_p - m|$



Gaussian



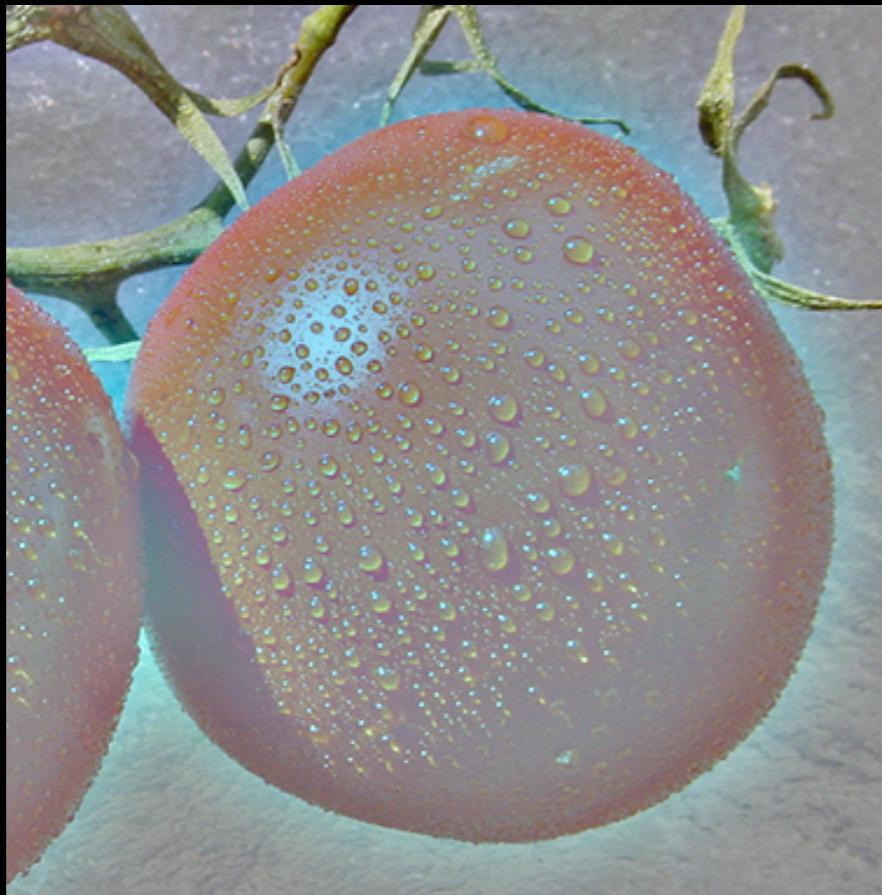
Median



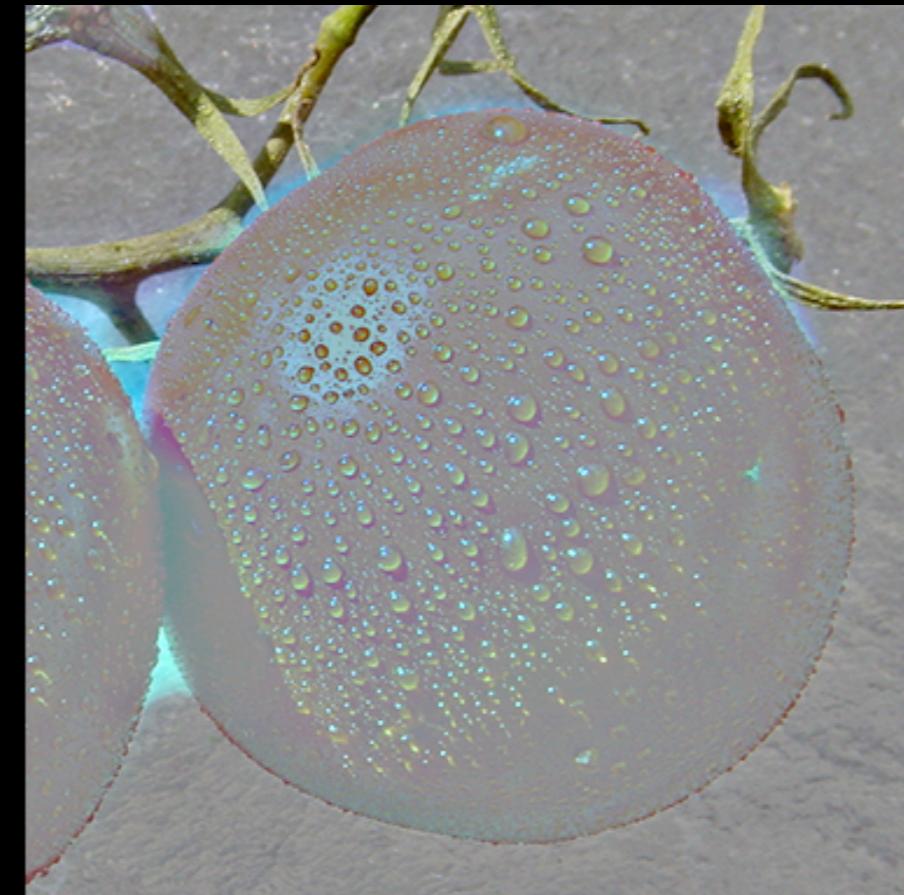
SIGGRAPH2006

# Why Median?

- Fundamental
  - Mean Minimizes  $|\sum I_p - m|$
  - Median Minimizes  $\sum |I_p - m|$



Gaussian



Median



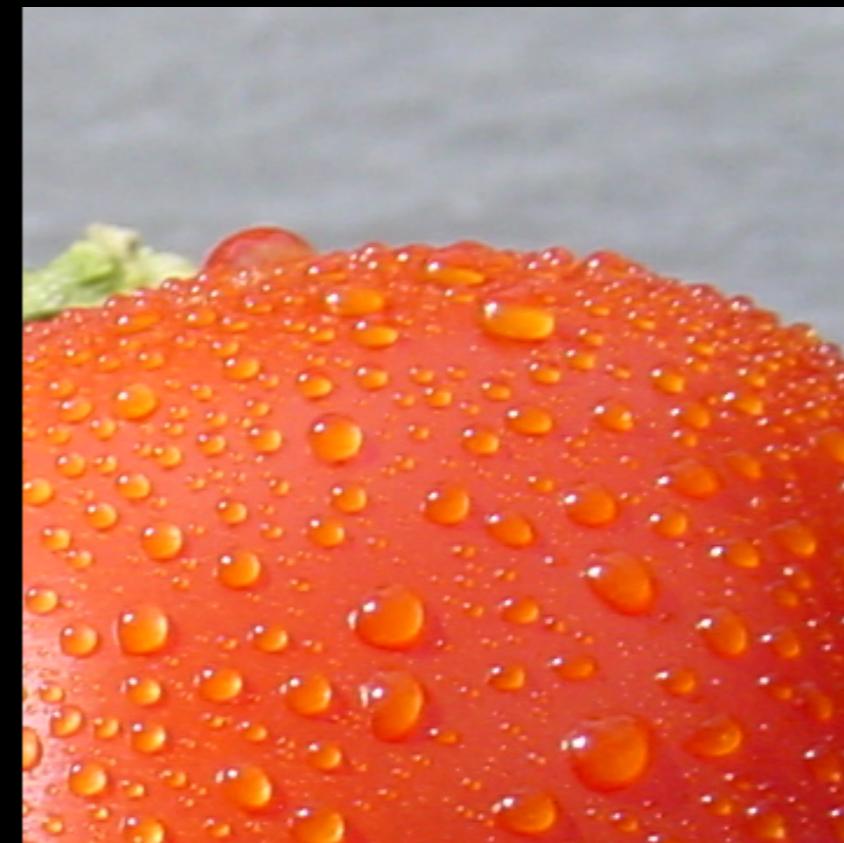
SIGGRAPH2006

# Why Median?

- Fundamental
  - Mean Minimizes  $|\sum I_p - m|$
  - Median Minimizes  $\sum |I_p - m|$



Gaussian



Median



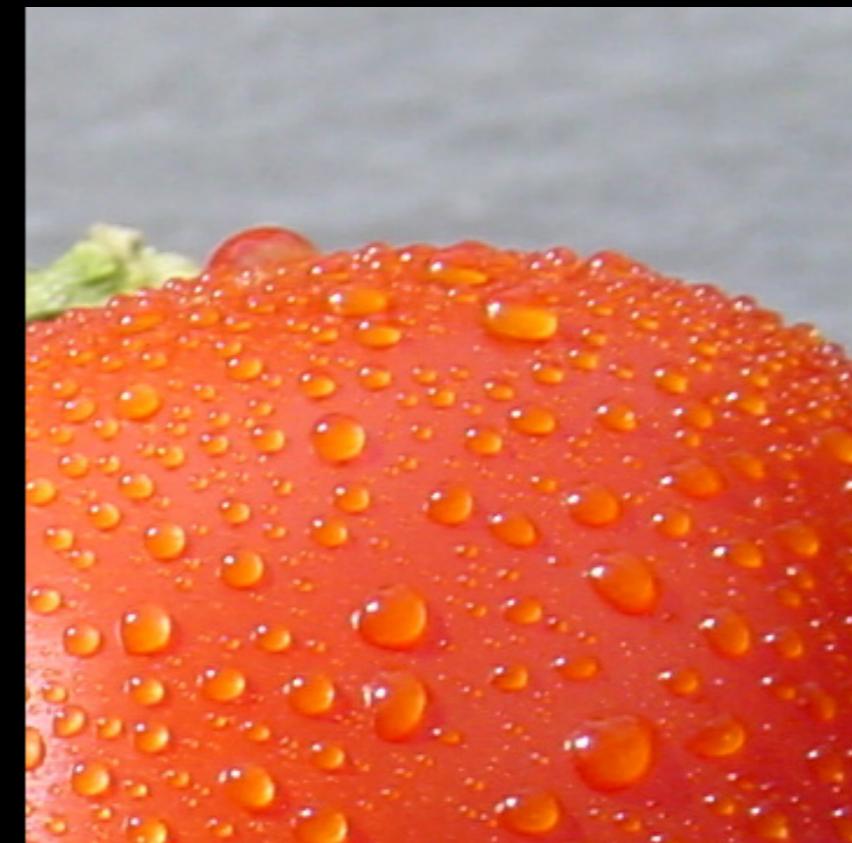
SIGGRAPH2006

# Why Median?

- Fundamental
  - Mean Minimizes  $|\sum I_p - m|$
  - Median Minimizes  $\sum |I_p - m|$



Gaussian



Median



SIGGRAPH2006

# Why Median?

- Fundamental
  - Mean Minimizes  $|\sum I_p - m|$
  - Median Minimizes  $\sum |I_p - m|$



Gaussian



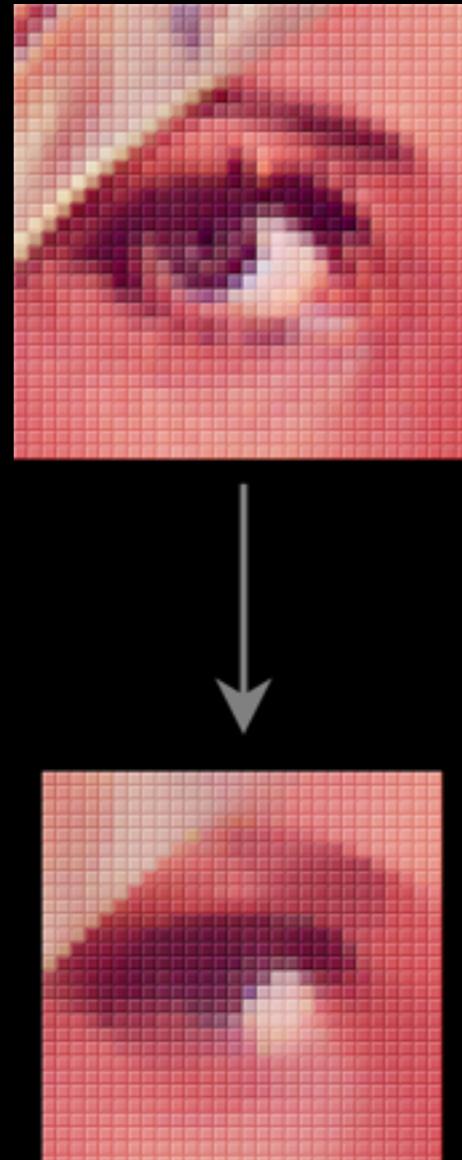
Median



SIGGRAPH2006

# Why Median?

- Fundamental
  - Mean Minimizes  $|\sum I_p - m|$
  - Median Minimizes  $\sum |I_p - m|$
- Challenging

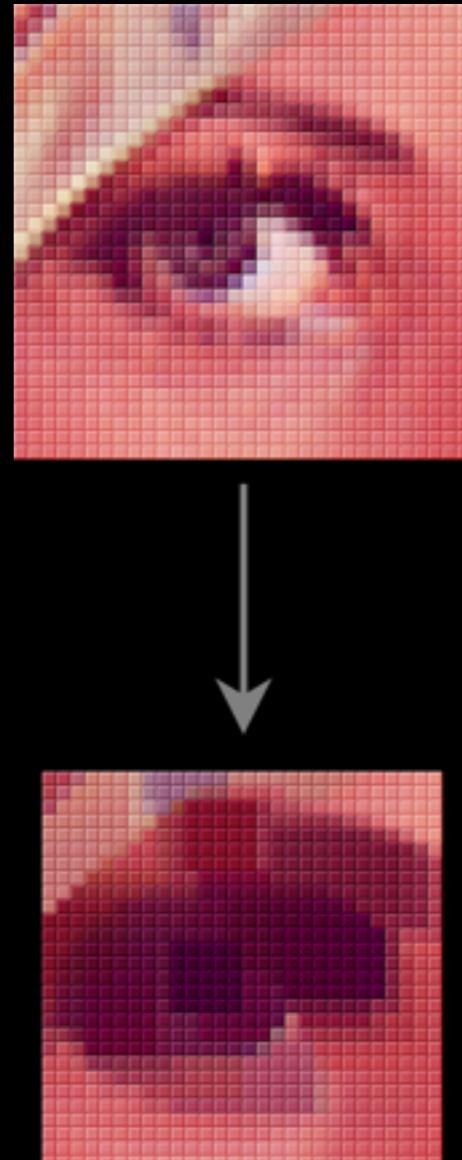




SIGGRAPH2006

# Why Median?

- Fundamental
  - Mean Minimizes  $|\sum I_p - m|$
  - Median Minimizes  $\sum |I_p - m|$
- Challenging
  - Minimum = 0<sup>th</sup> Percentile = Easy

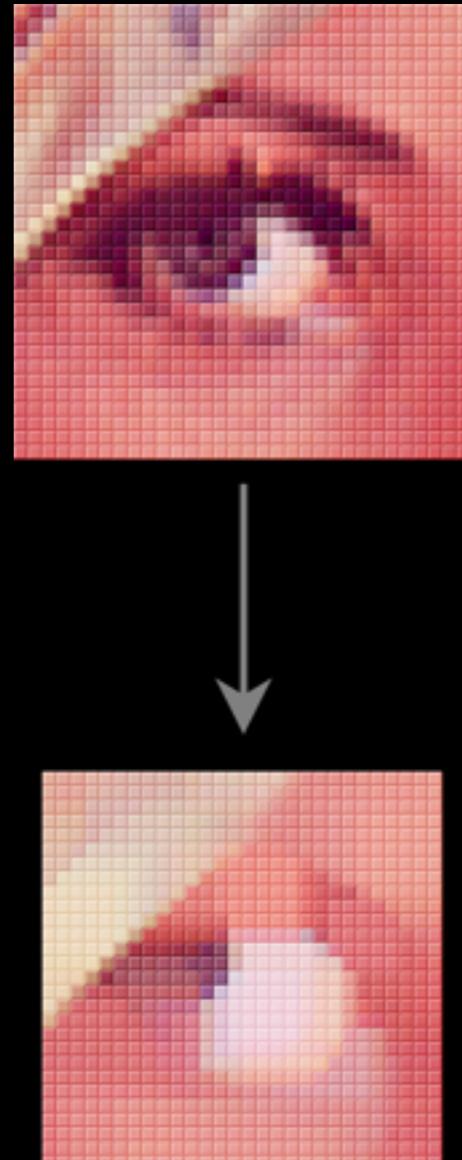




SIGGRAPH2006

# Why Median?

- Fundamental
  - Mean Minimizes  $\left| \sum I_p - m \right|$
  - Median Minimizes  $\sum |I_p - m|$
- Challenging
  - Minimum = 0<sup>th</sup> Percentile = Easy
  - Maximum = 100<sup>th</sup> Percentile = Easy

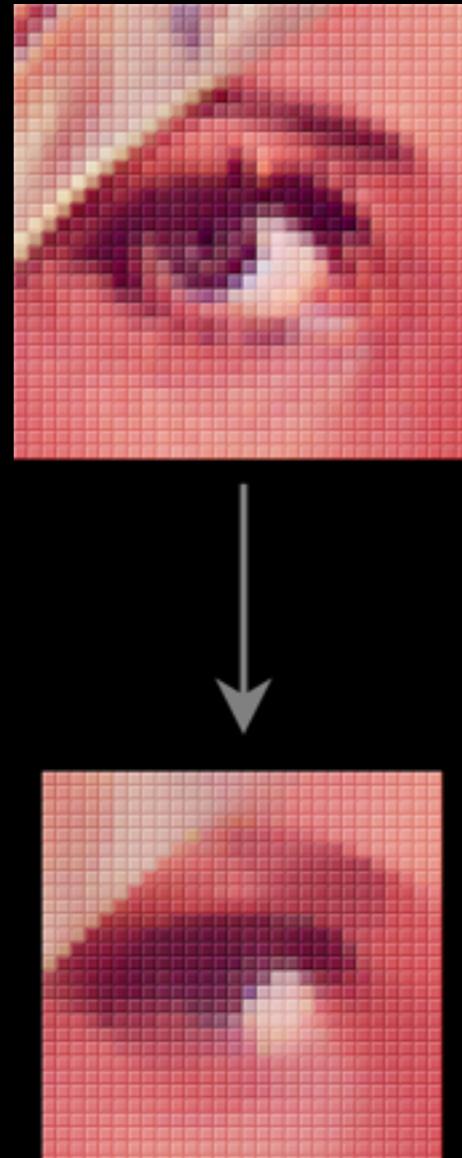




SIGGRAPH2006

# Why Median?

- Fundamental
  - Mean Minimizes  $\left| \sum I_p - m \right|$
  - Median Minimizes  $\sum |I_p - m|$
- Challenging
  - Minimum = 0<sup>th</sup> Percentile = Easy
  - Maximum = 100<sup>th</sup> Percentile = Easy
  - Median = 50<sup>th</sup> Percentile = Hard!

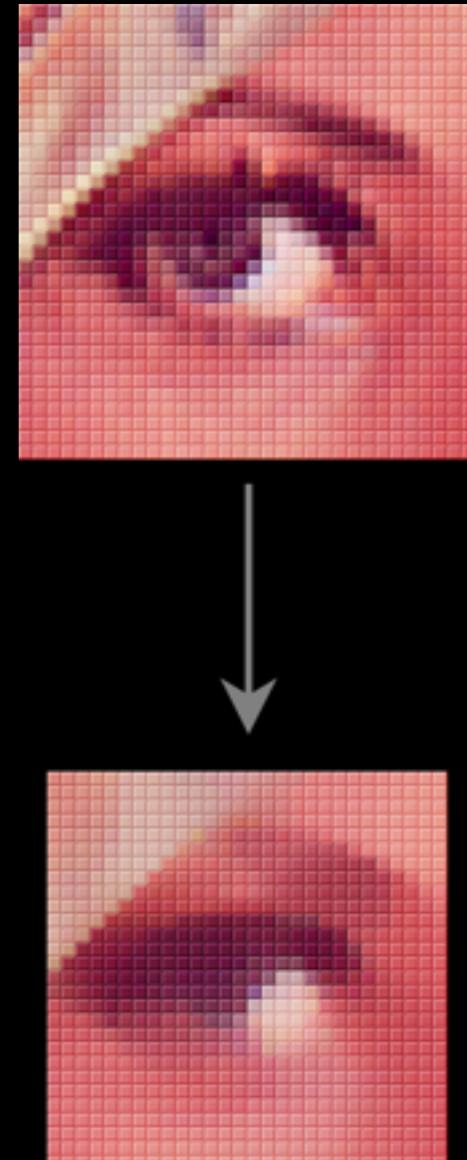




SIGGRAPH2006

# Why Median?

- Fundamental
  - Mean Minimizes  $\left| \sum I_p - m \right|$
  - Median Minimizes  $\sum |I_p - m|$
- Challenging
  - Minimum = 0<sup>th</sup> Percentile = Easy
  - Maximum = 100<sup>th</sup> Percentile = Easy
  - Median = 50<sup>th</sup> Percentile = Hard!
- Versatile



# Outline



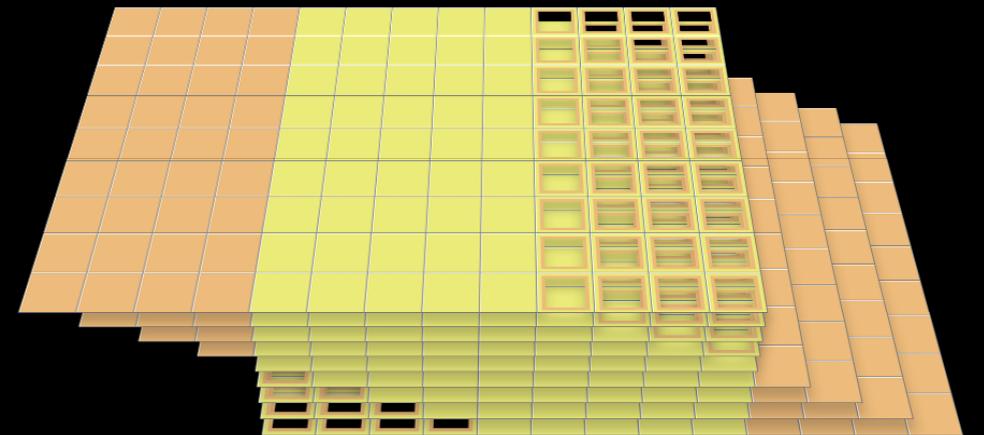
SIGGRAPH2006



SIGGRAPH2006

# Outline

- 8-bit Median

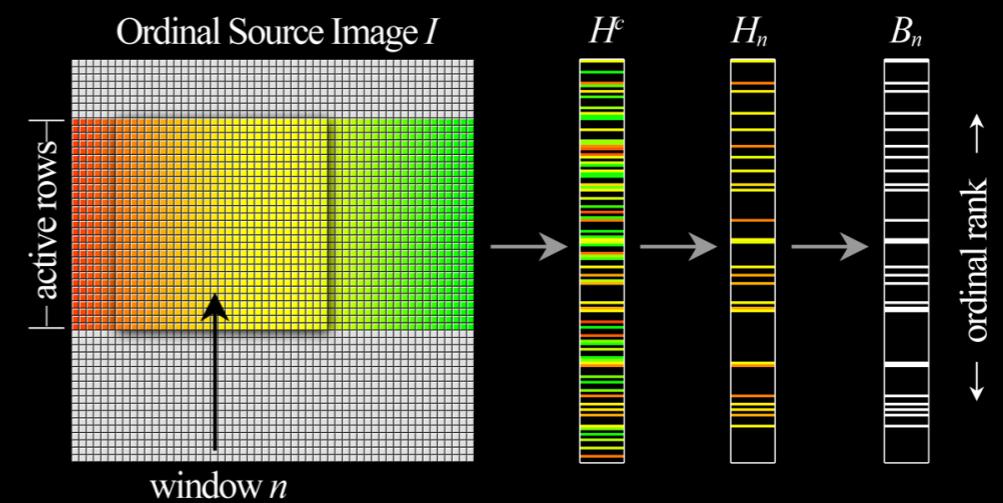
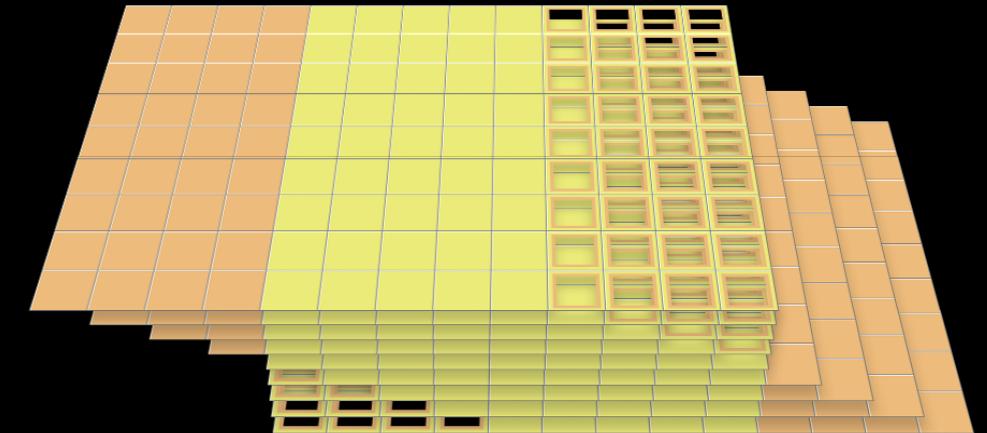




SIGGRAPH2006

# Outline

- 8-bit Median
- High-Precision Median

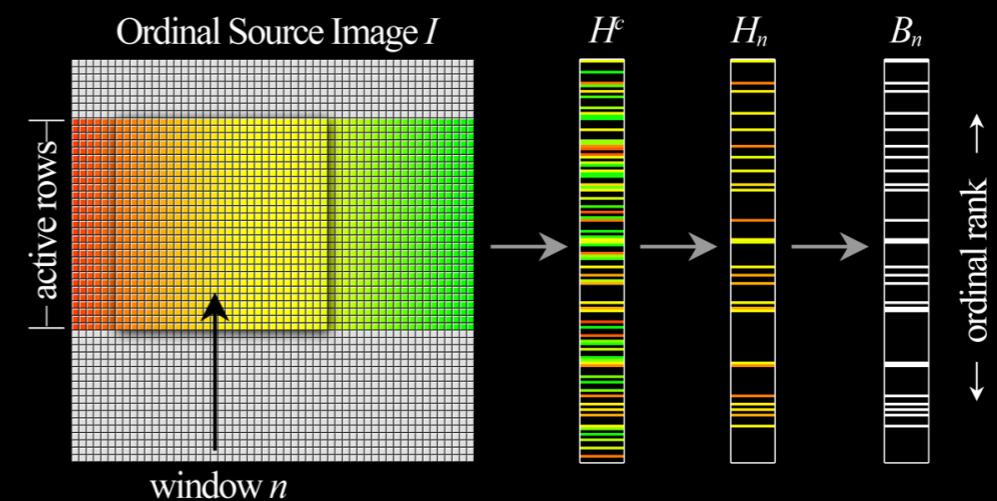
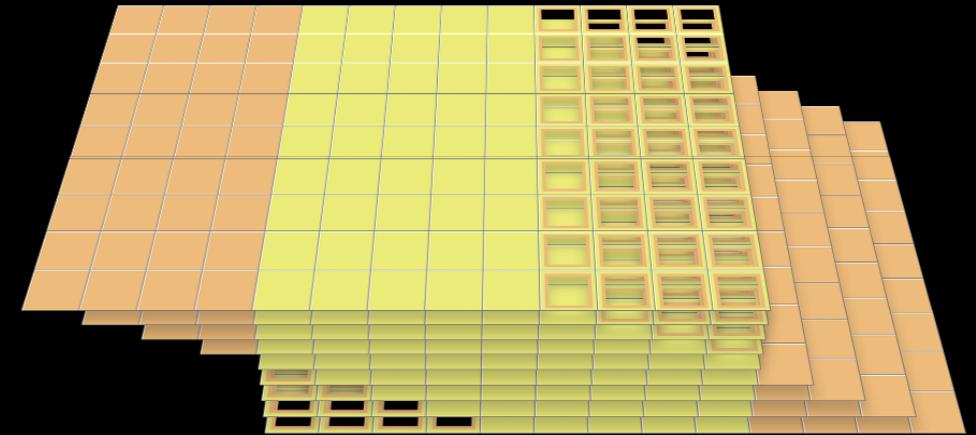


# Outline



SIGGRAPH2006

- 8-bit Median
- High-Precision Median
- Bilateral Filter

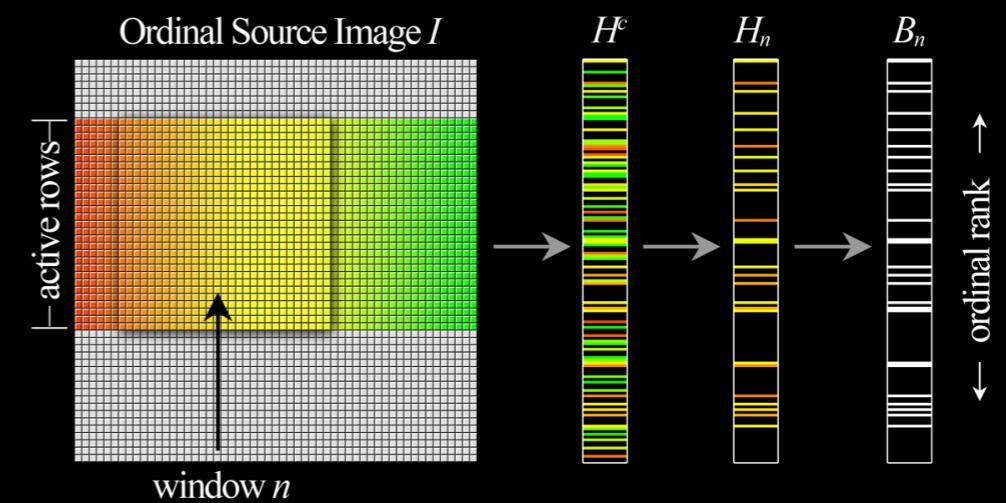
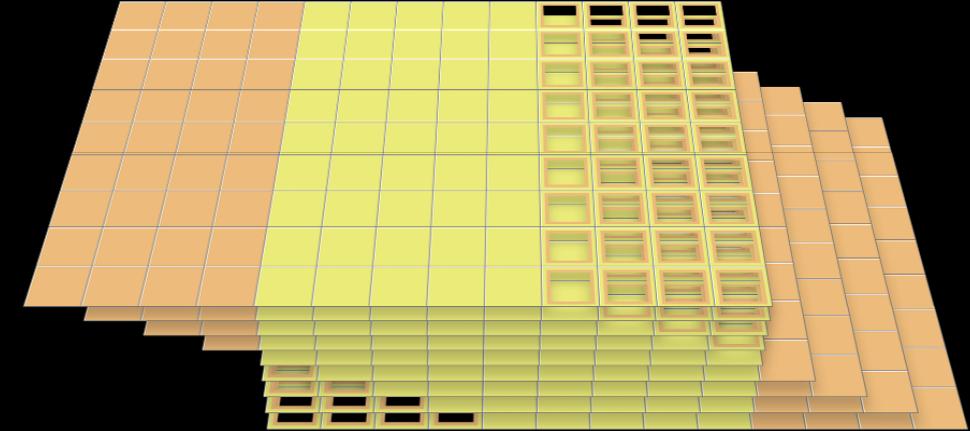


# Outline



SIGGRAPH2006

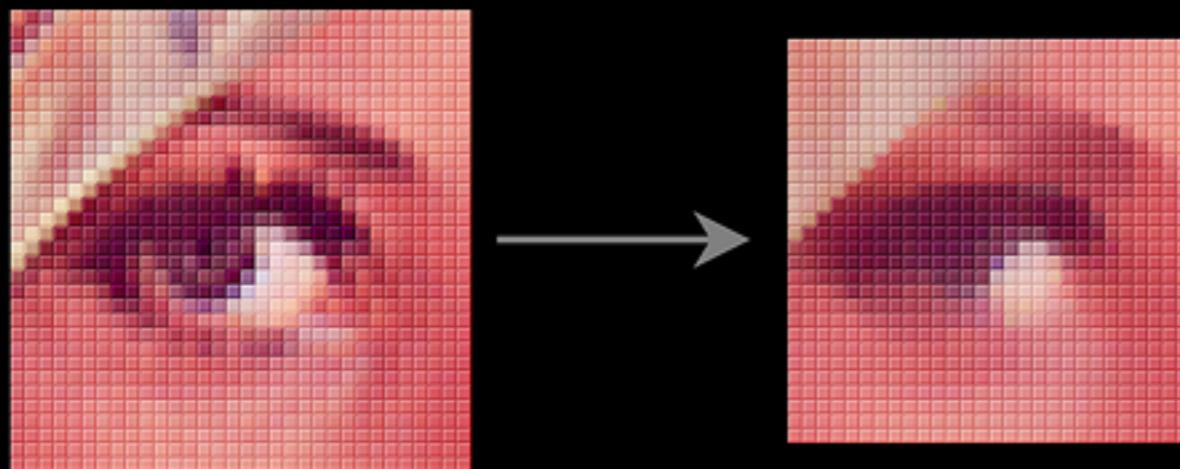
- 8-bit Median
- High-Precision Median
- Bilateral Filter
- Creative Applications





SIGGRAPH2006

# Standard 8-Bit $O(r)$ Algorithm

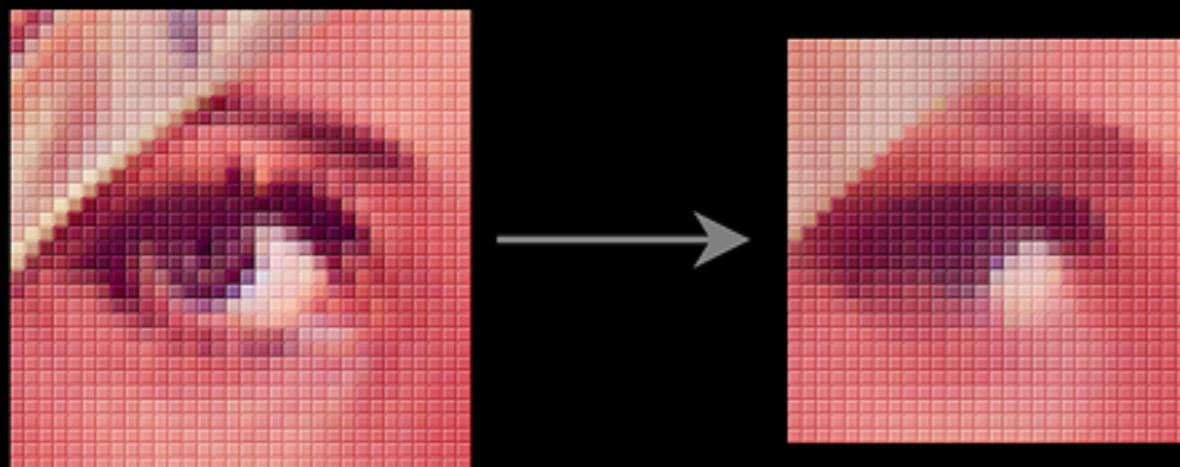
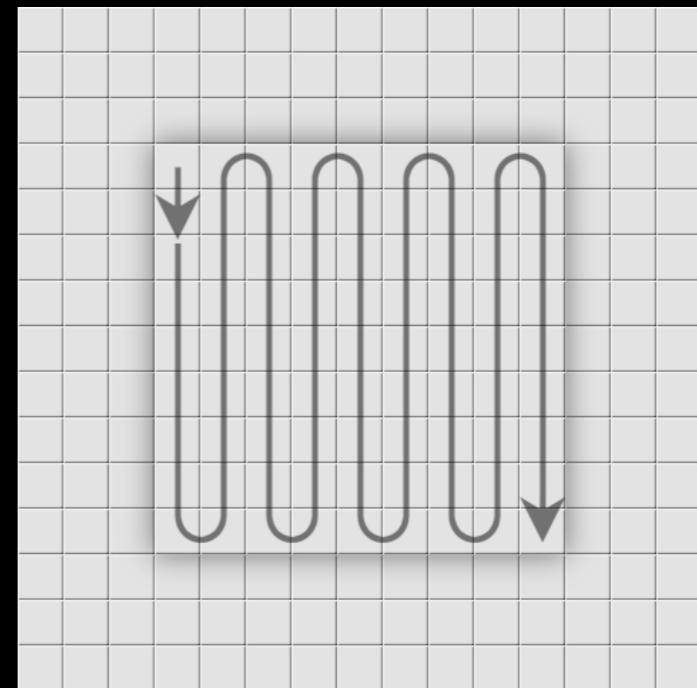




SIGGRAPH2006

# Standard 8-Bit $O(r)$ Algorithm

- One Column at a Time

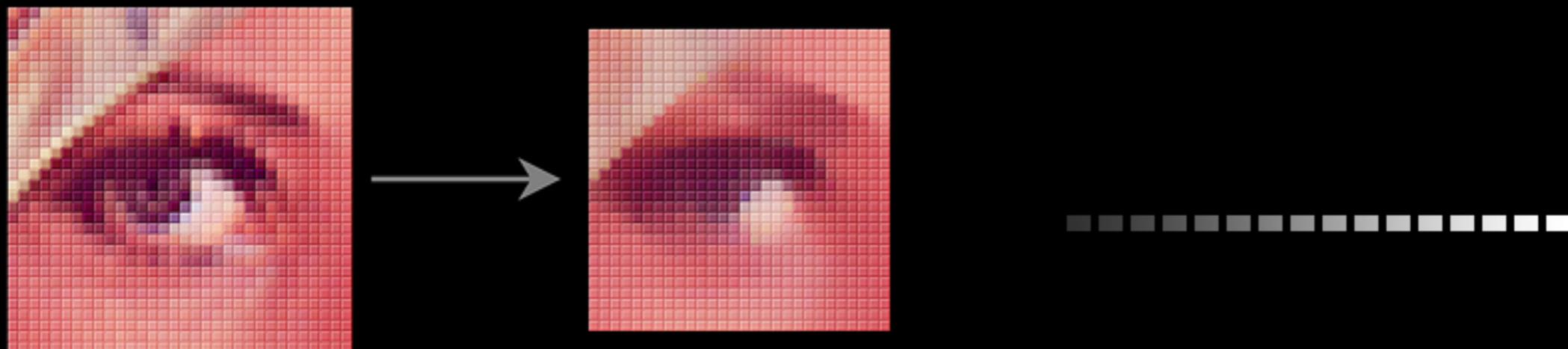
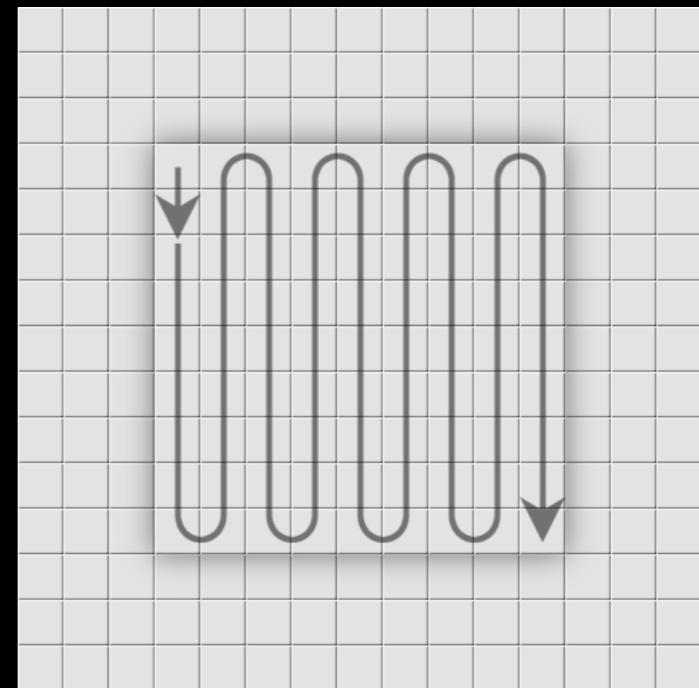




SIGGRAPH2006

# Standard 8-Bit $O(r)$ Algorithm

- One Column at a Time
- 256-Element Histogram  $H$

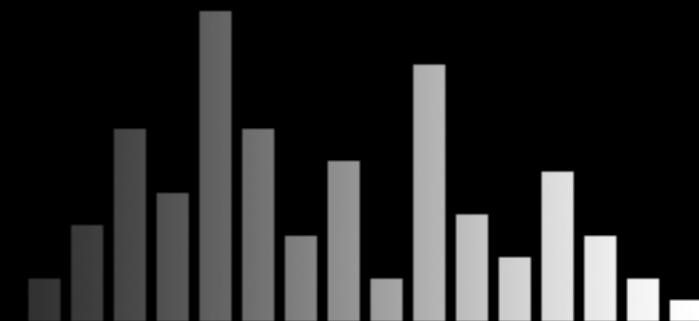
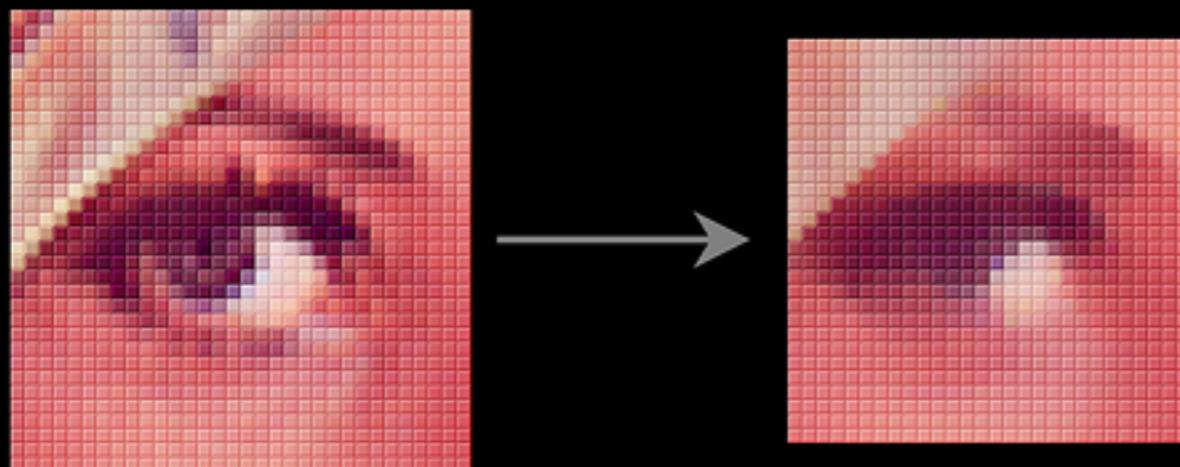
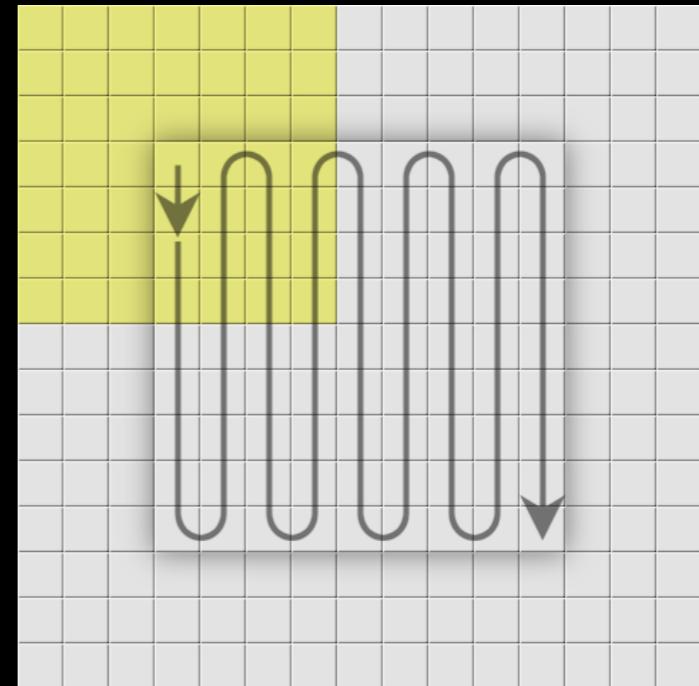




SIGGRAPH2006

# Standard 8-Bit $O(r)$ Algorithm

- One Column at a Time
- 256-Element Histogram  $H$ 
  - Initialize  $H$  to Top Left

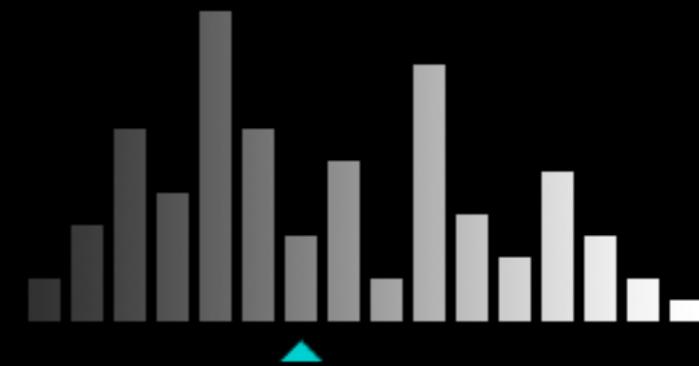
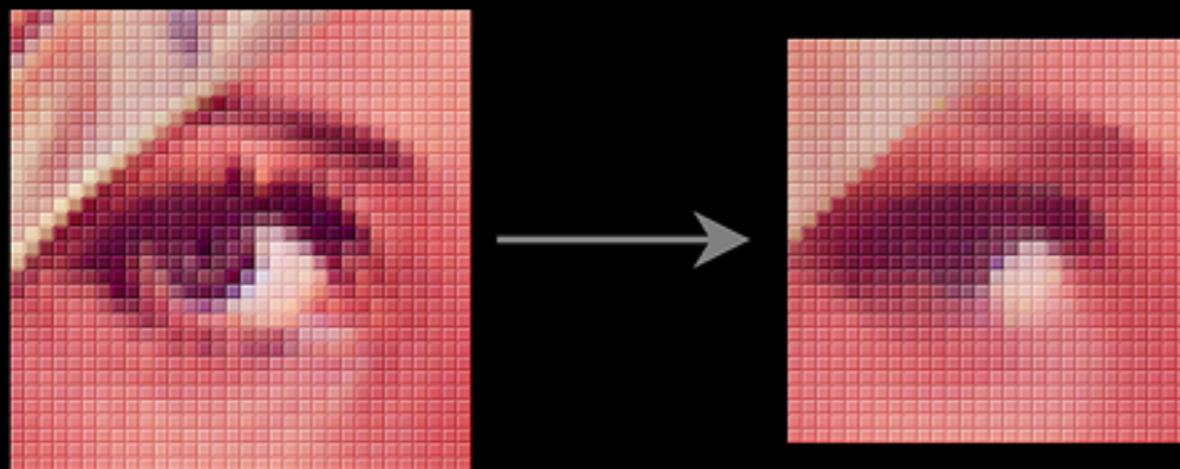
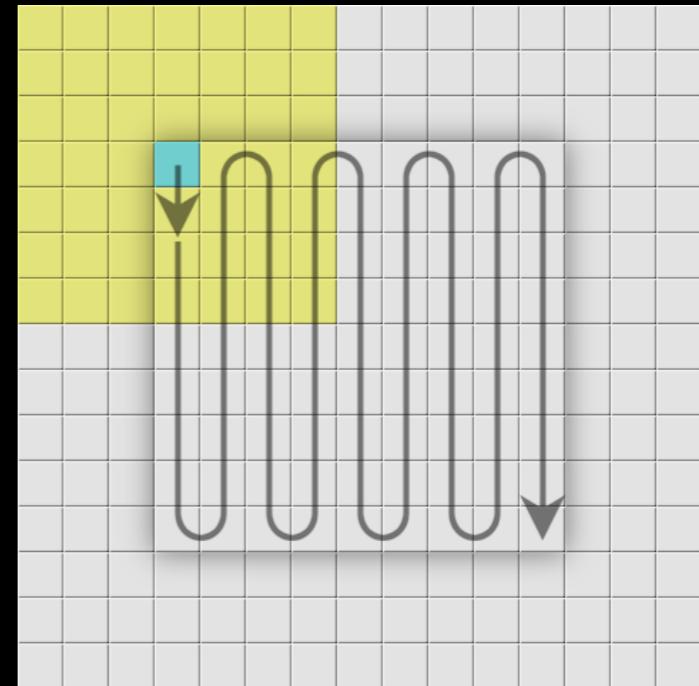




SIGGRAPH2006

# Standard 8-Bit $O(r)$ Algorithm

- One Column at a Time
- 256-Element Histogram  $H$ 
  - Initialize  $H$  to Top Left
  - Extract Median from  $H$

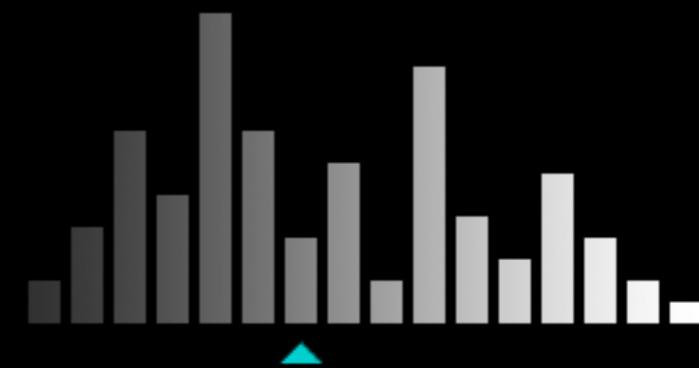
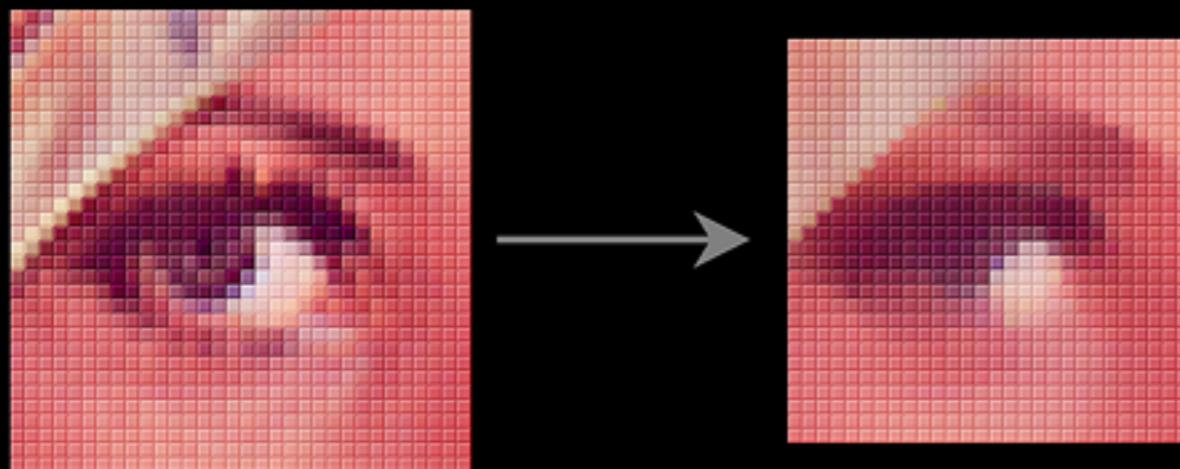
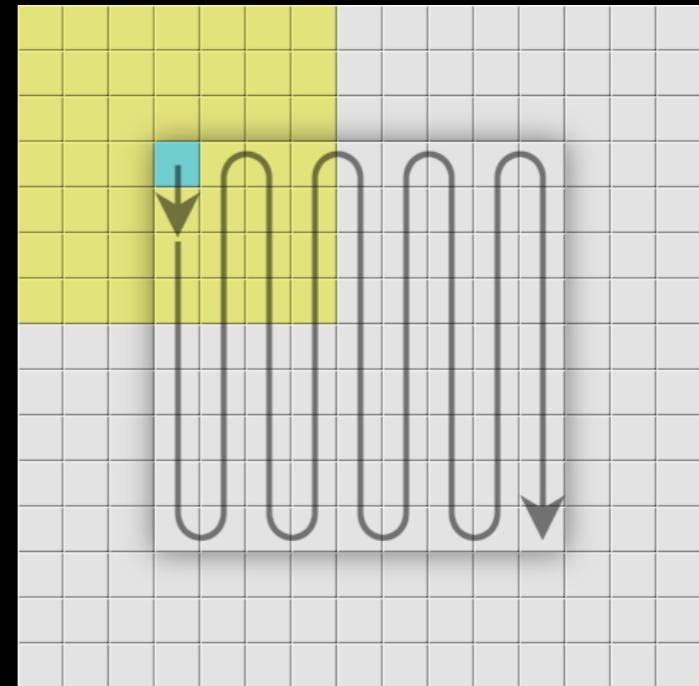




SIGGRAPH2006

# Standard 8-Bit $O(r)$ Algorithm

- One Column at a Time
- 256-Element Histogram  $H$ 
  - Initialize  $H$  to Top Left
  - Extract Median from  $H$
  - Slide Window, Update  $H$

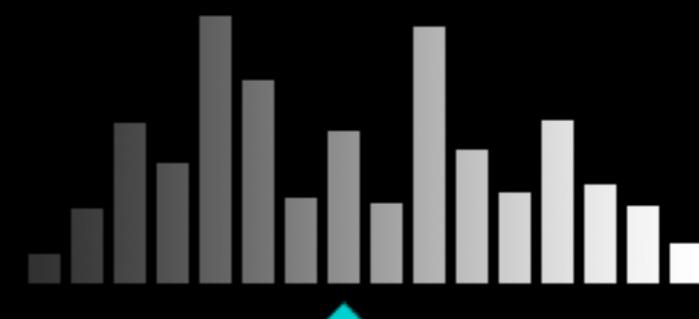
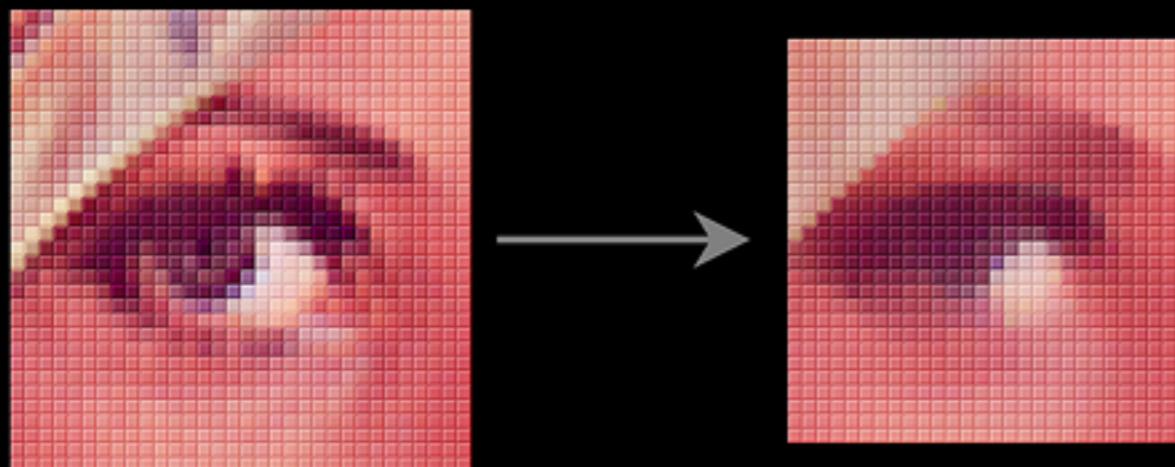
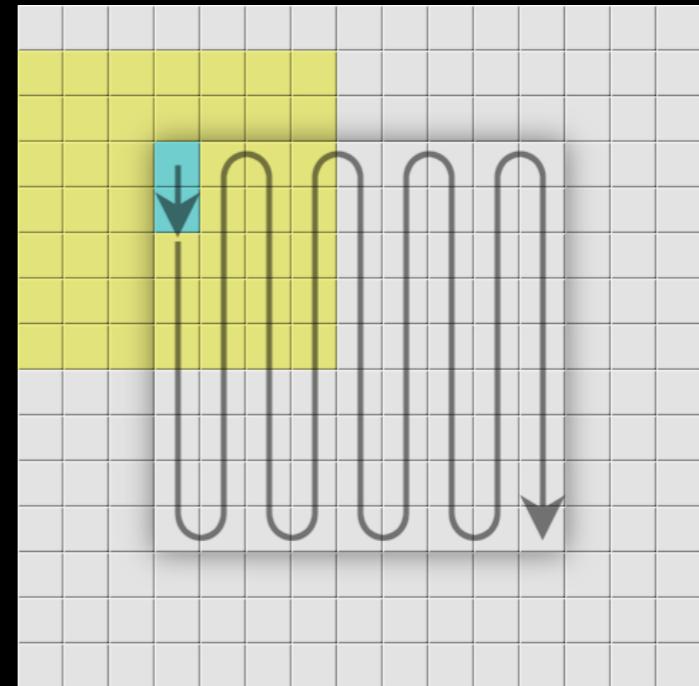




SIGGRAPH2006

# Standard 8-Bit $O(r)$ Algorithm

- One Column at a Time
- 256-Element Histogram  $H$ 
  - Initialize  $H$  to Top Left
  - Extract Median from  $H$
  - Slide Window, Update  $H$

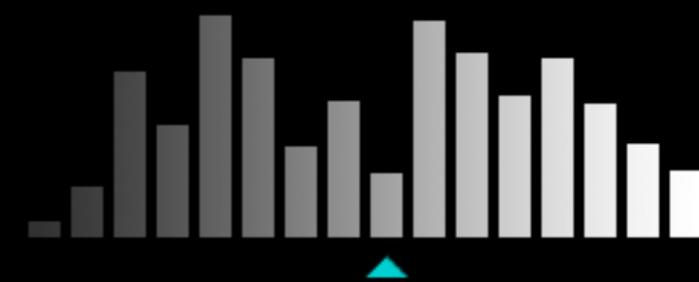
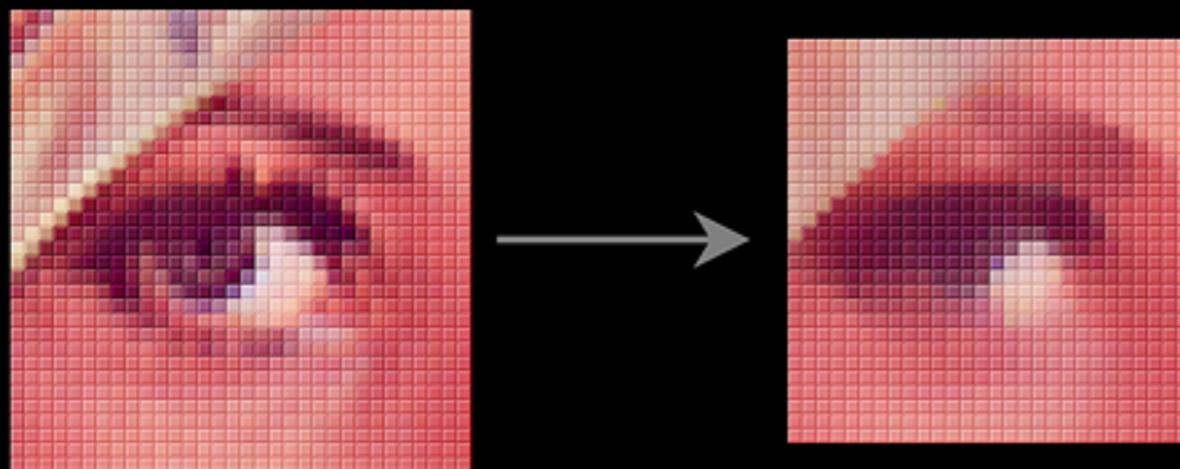
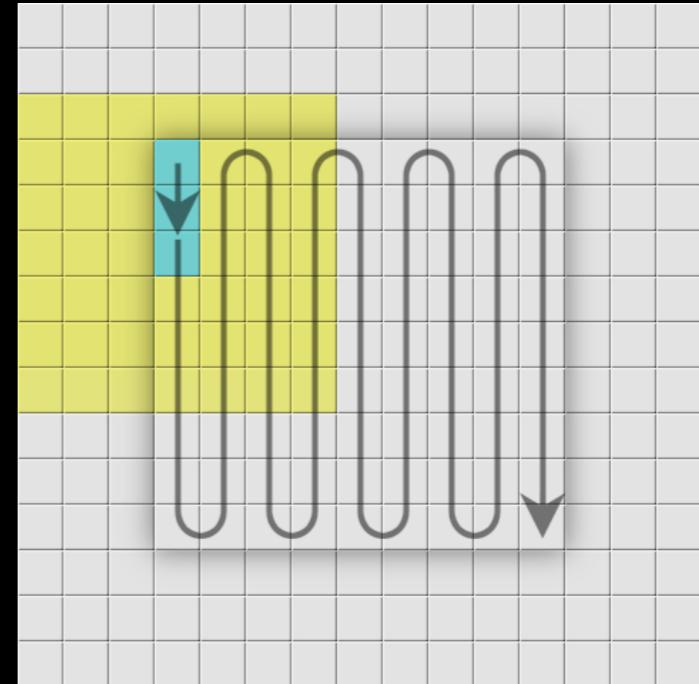




SIGGRAPH2006

# Standard 8-Bit $O(r)$ Algorithm

- One Column at a Time
- 256-Element Histogram  $H$ 
  - Initialize  $H$  to Top Left
  - Extract Median from  $H$
  - Slide Window, Update  $H$

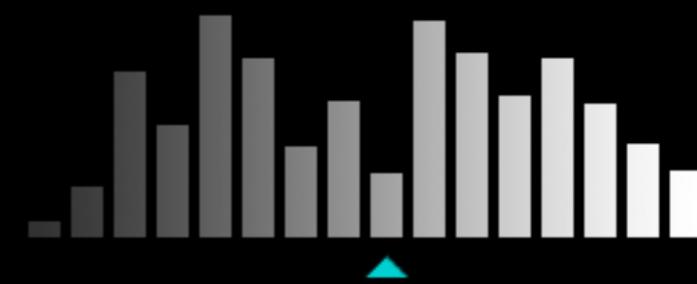
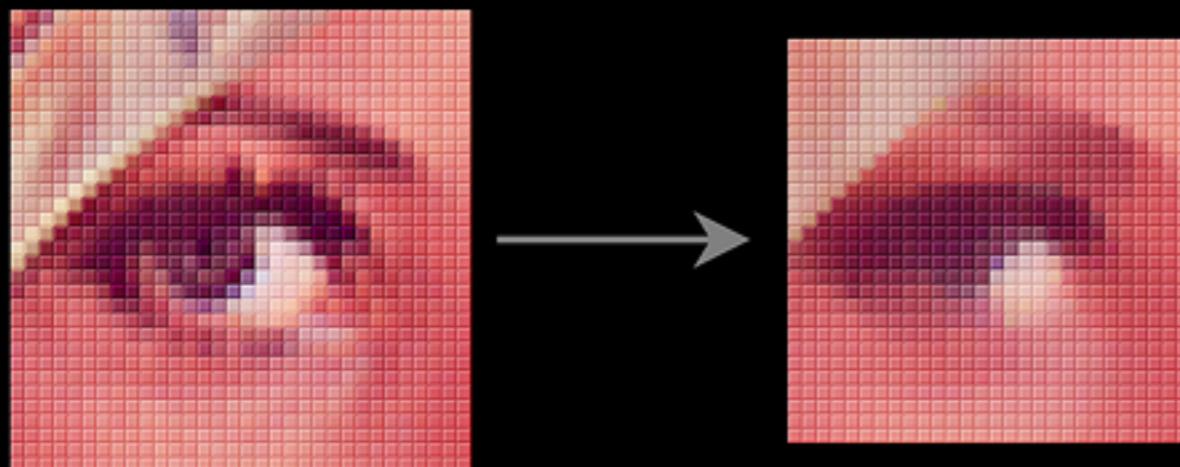
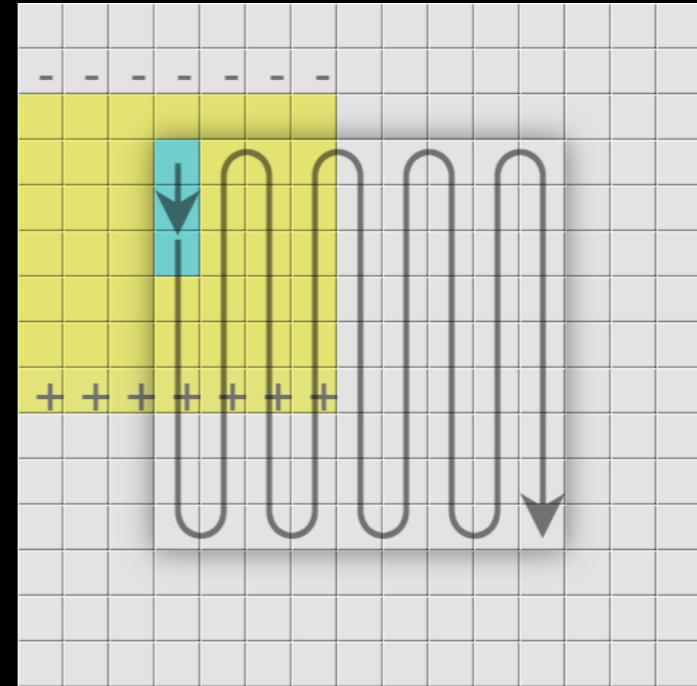




SIGGRAPH2006

# Standard 8-Bit $O(r)$ Algorithm

- One Column at a Time
- 256-Element Histogram  $H$ 
  - Initialize  $H$  to Top Left
  - Extract Median from  $H$
  - Slide Window, Update  $H$
- Redundant Computation



# Parallel $O(r)$ Algorithm

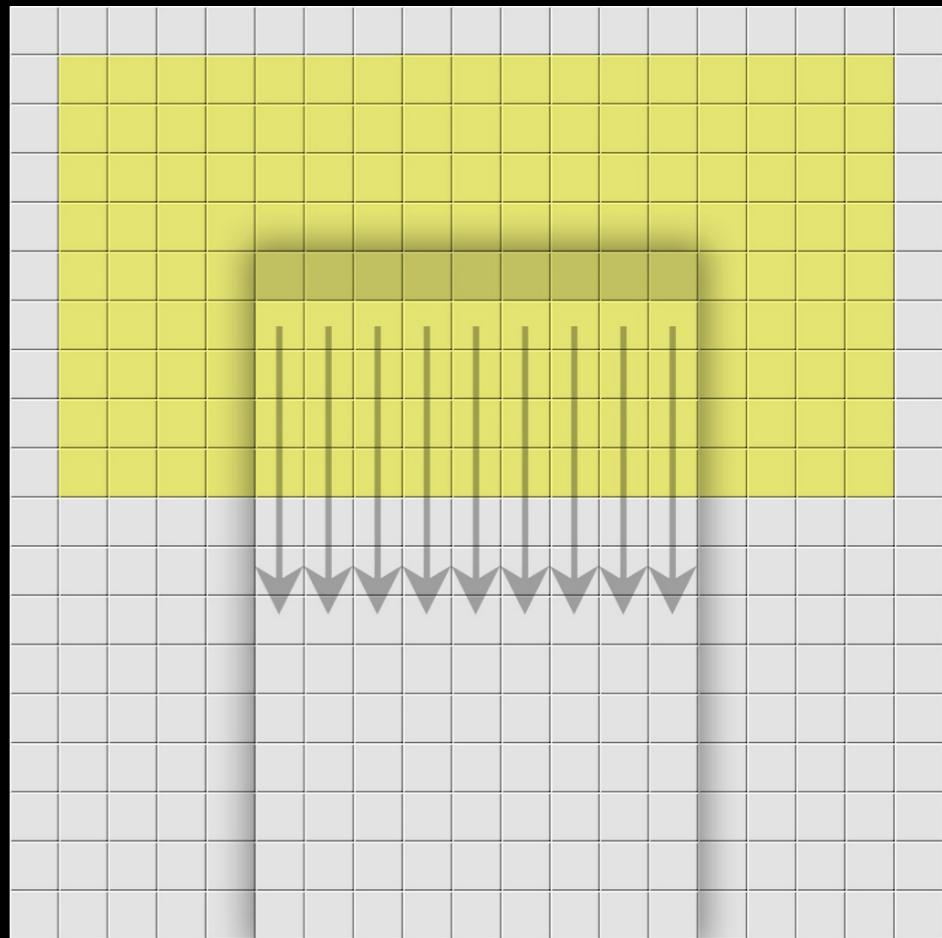


SIGGRAPH2006

# Parallel $O(r)$ Algorithm



- E.g., Nine Columns at Once

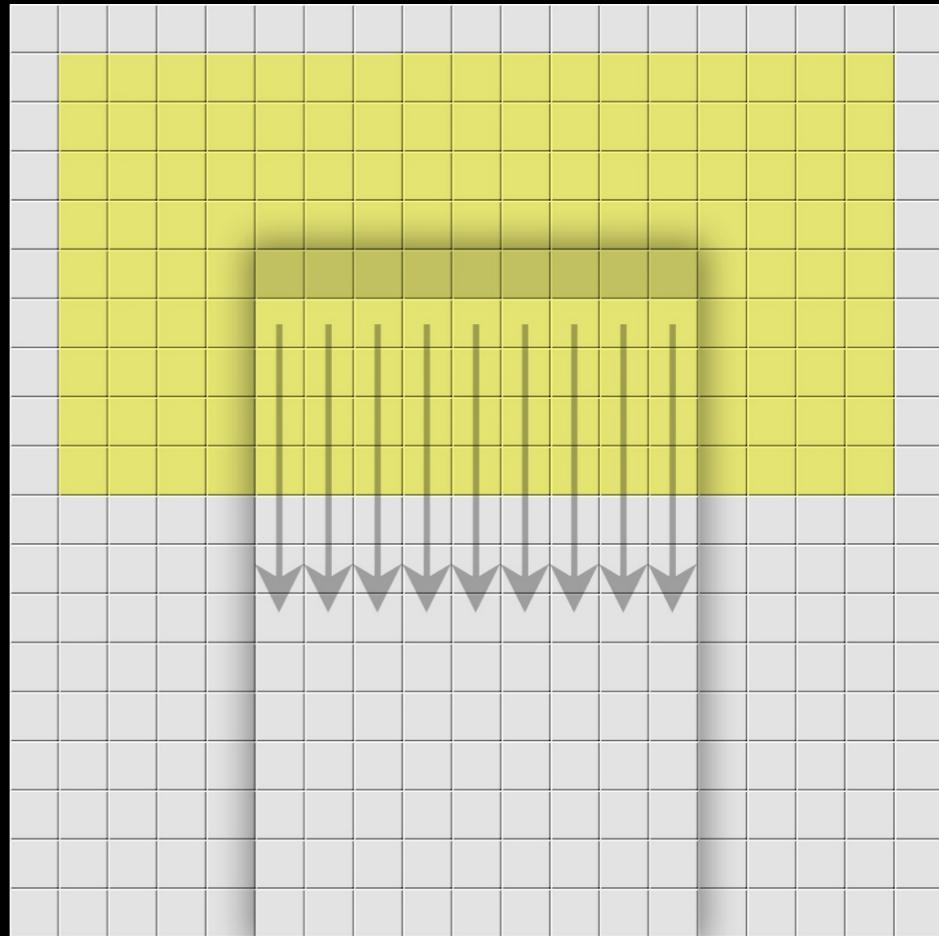
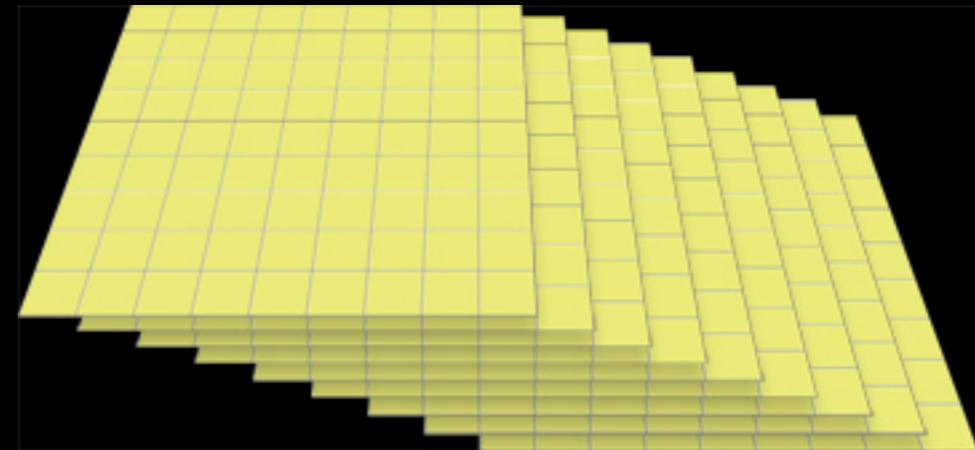




SIGGRAPH2006

# Parallel $O(r)$ Algorithm

- E.g., Nine Columns at Once
- 9 Columns  $\Leftrightarrow$  9 Windows

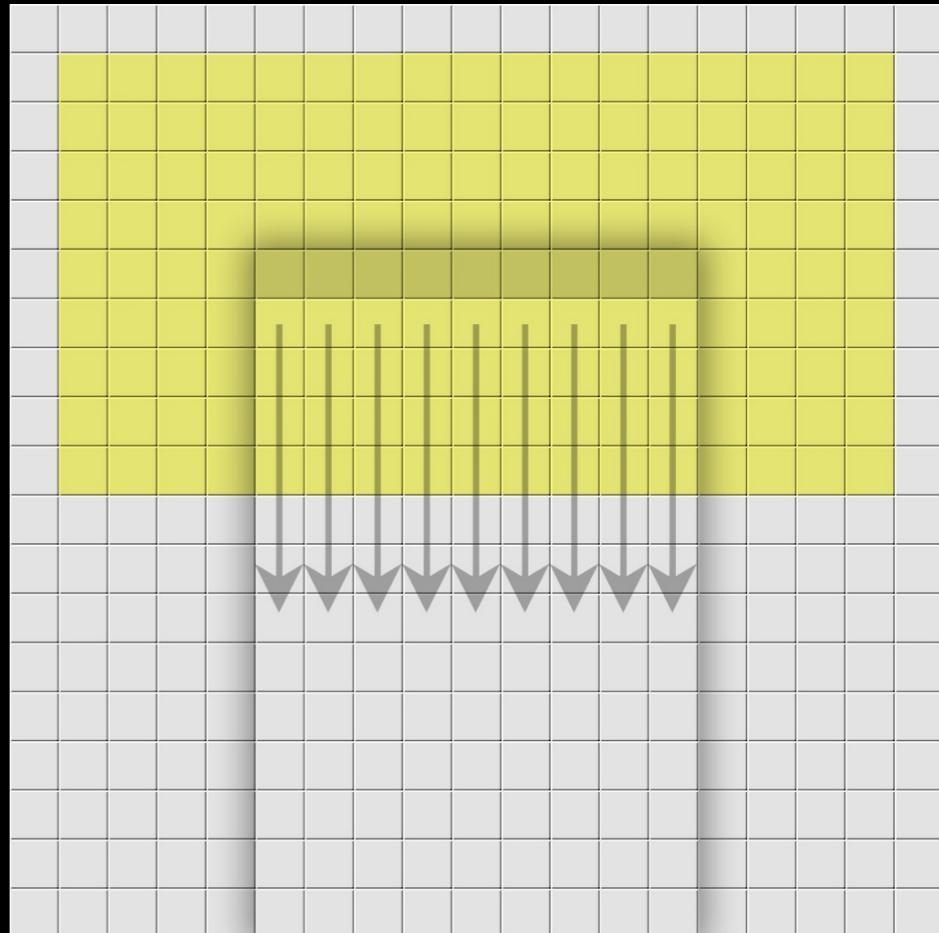
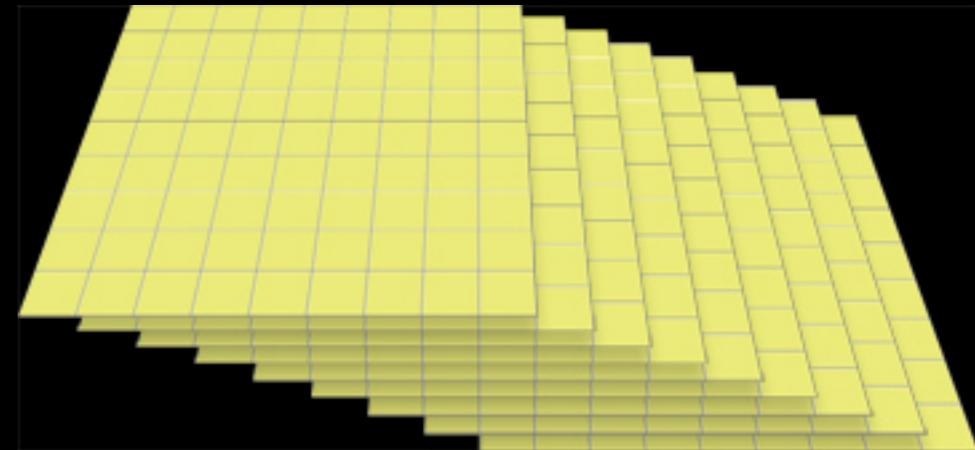




SIGGRAPH2006

# Parallel $O(r)$ Algorithm

- E.g., Nine Columns at Once
- 9 Columns  $\Leftrightarrow$  9 Windows
- 9 Windows  $\Leftrightarrow$  9 Histograms

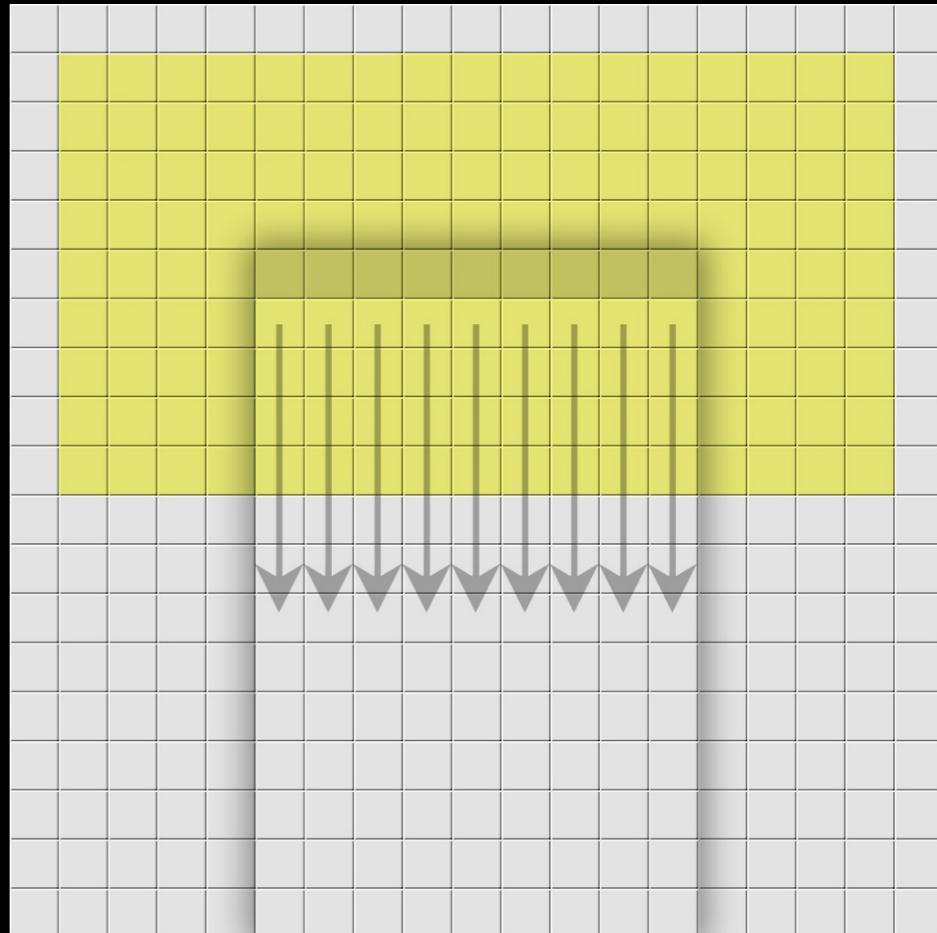
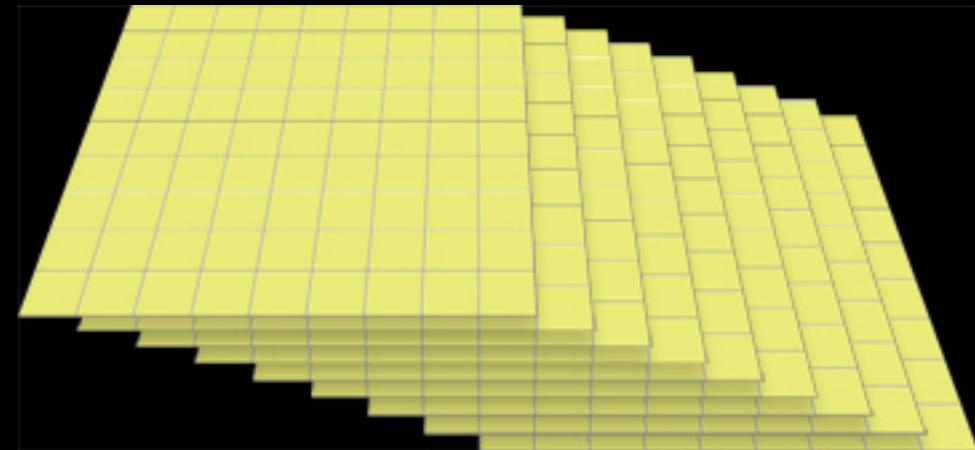




SIGGRAPH2006

# Parallel $O(r)$ Algorithm

- E.g., Nine Columns at Once
- 9 Columns  $\Leftrightarrow$  9 Windows
- 9 Windows  $\Leftrightarrow$  9 Histograms
- Histograms are **Distributive**

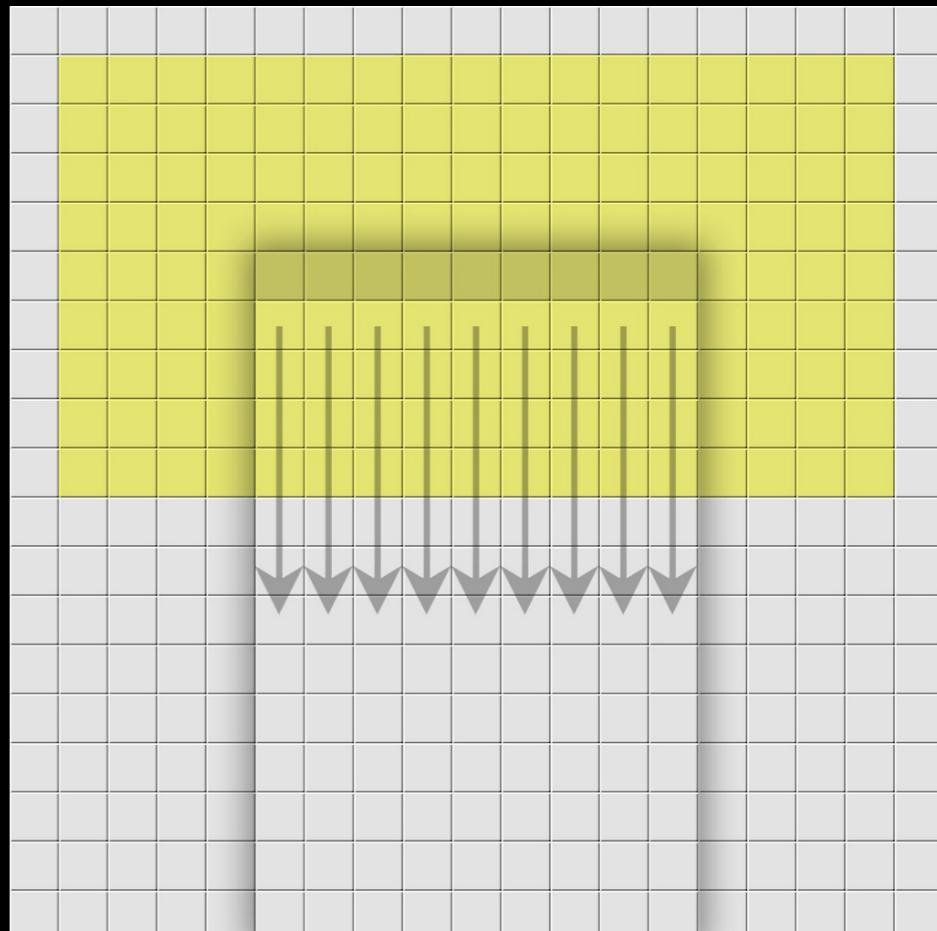
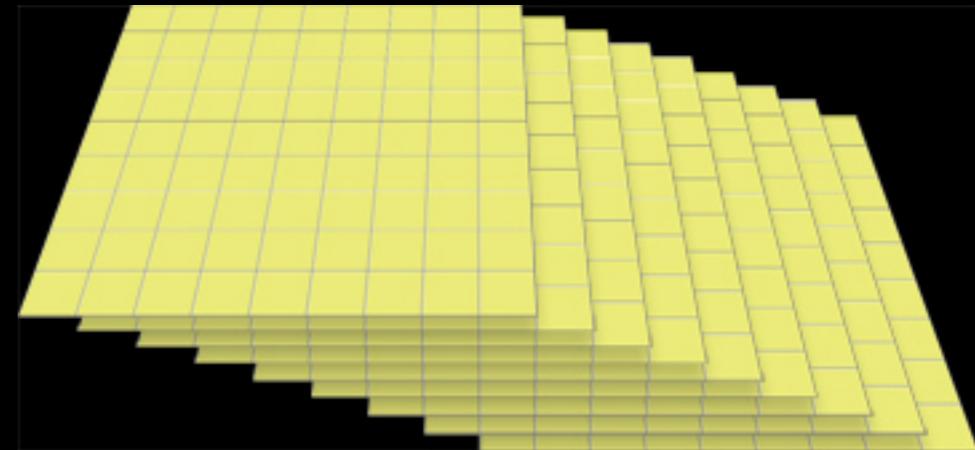




SIGGRAPH2006

# Parallel $O(r)$ Algorithm

- E.g., Nine Columns at Once
- 9 Columns  $\Leftrightarrow$  9 Windows
- 9 Windows  $\Leftrightarrow$  9 Histograms
- Histograms are **Distributive**
  - $H_{A \cup B}[v] = H_A[v] + H_B[v]$

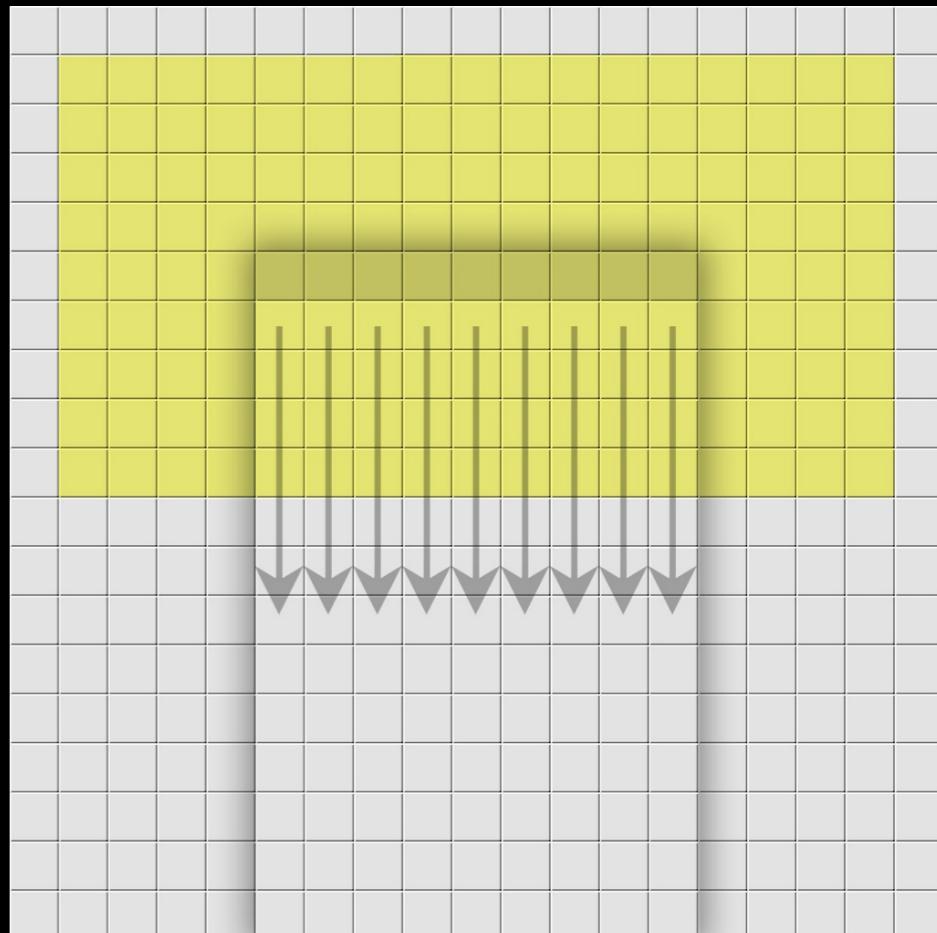
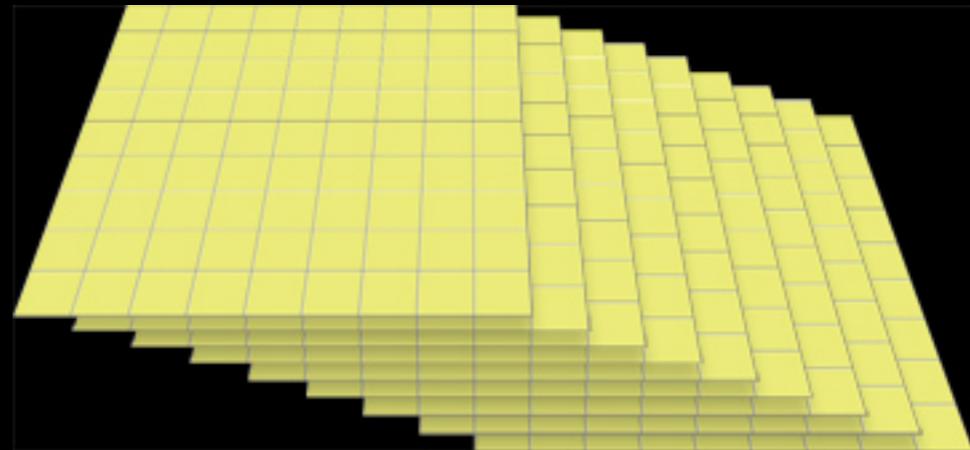




SIGGRAPH2006

# Parallel $O(r)$ Algorithm

- E.g., Nine Columns at Once
- 9 Columns  $\Leftrightarrow$  9 Windows
- 9 Windows  $\Leftrightarrow$  9 Histograms
- Histograms are **Distributive**
  - $H_{A \cup B}[v] = H_A[v] + H_B[v]$
  - $H_n =$

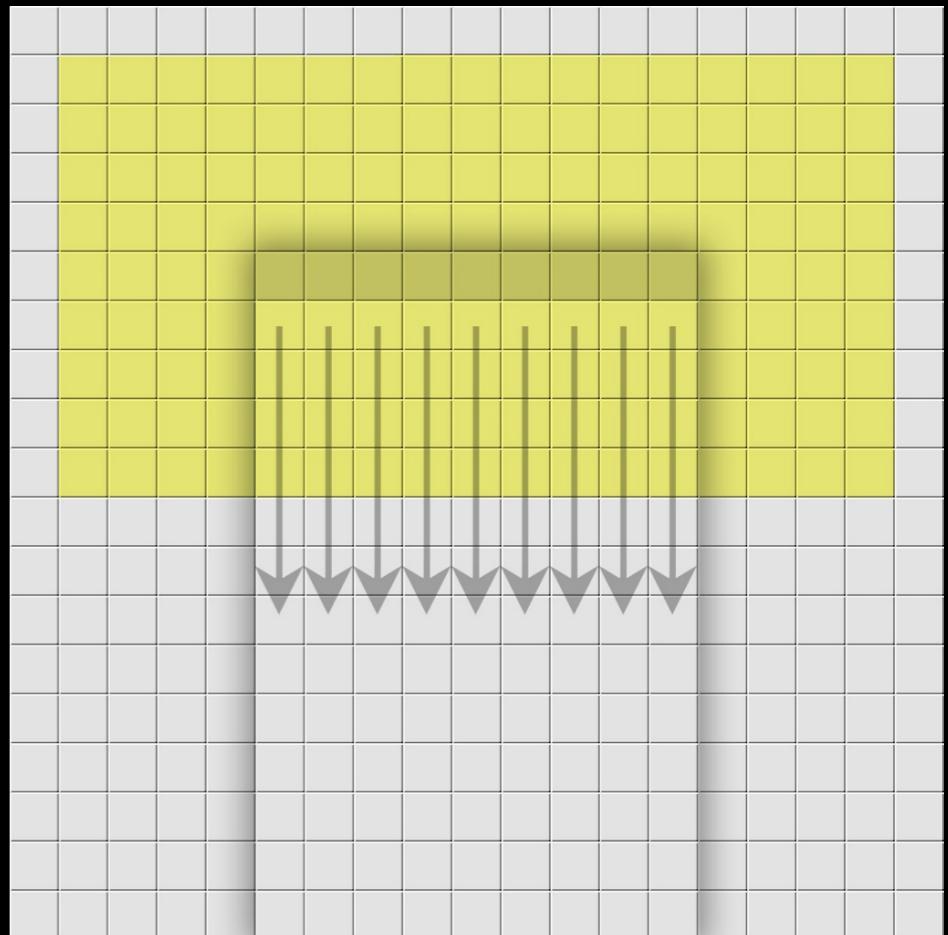
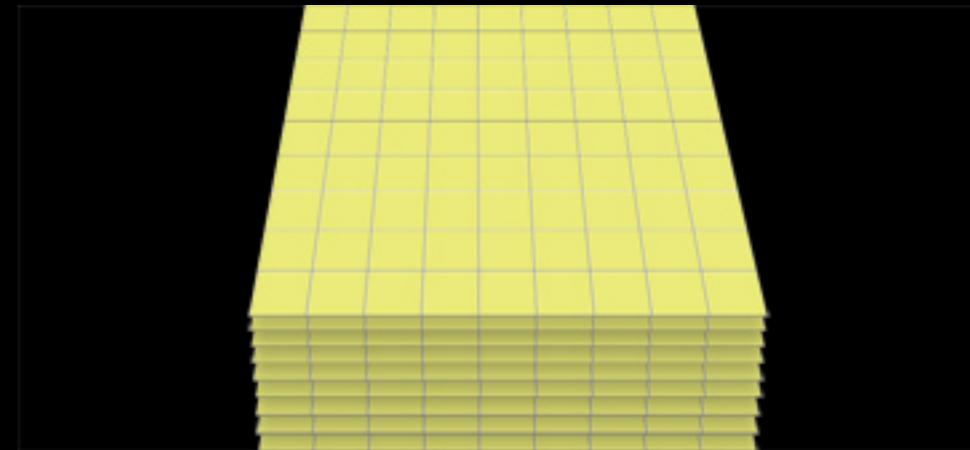


# Parallel $O(r)$ Algorithm



SIGGRAPH2006

- E.g., Nine Columns at Once
- 9 Columns  $\Leftrightarrow$  9 Windows
- 9 Windows  $\Leftrightarrow$  9 Histograms
- Histograms are **Distributive**
  - $H_{A \cup B}[v] = H_A[v] + H_B[v]$
  - $H_n = H_c$

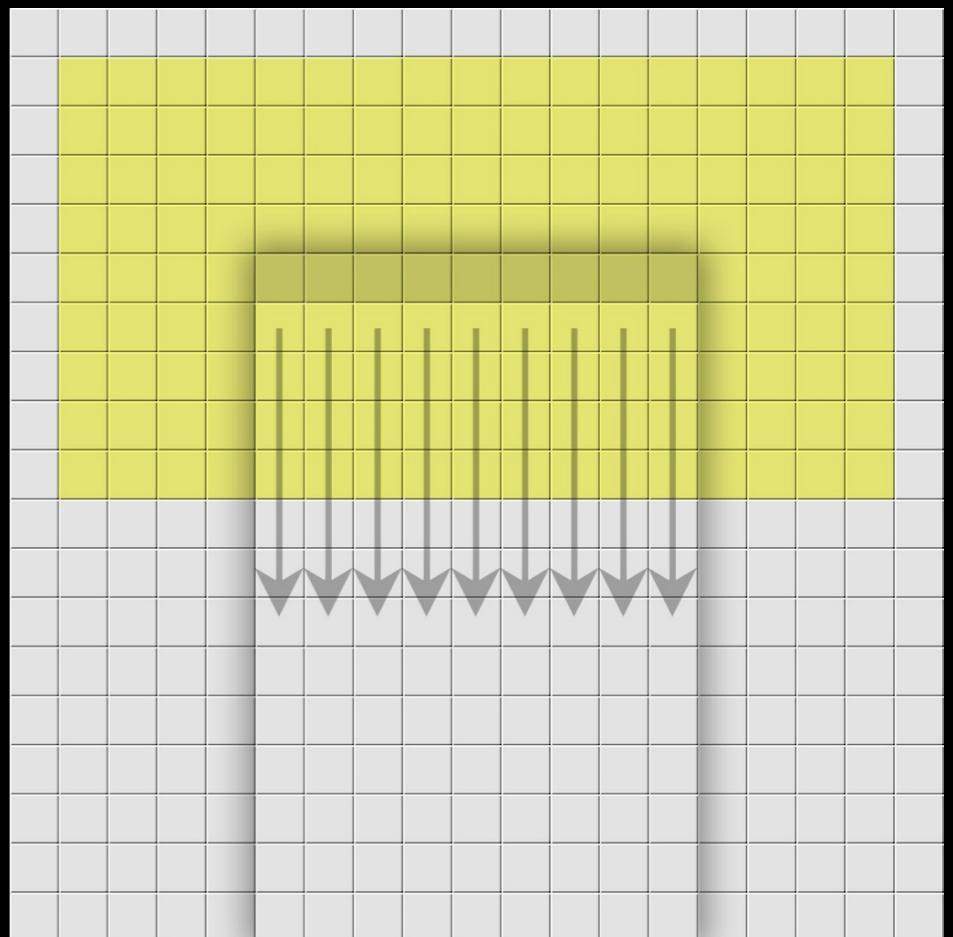
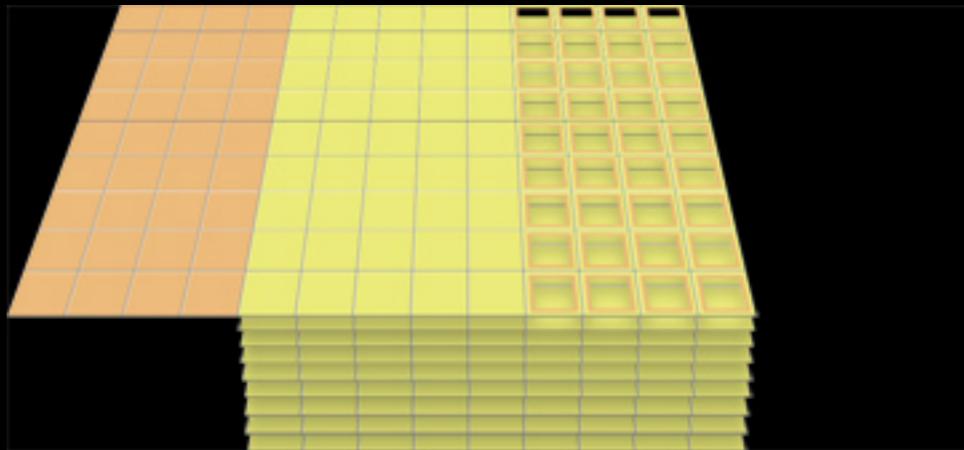




SIGGRAPH2006

# Parallel $O(r)$ Algorithm

- E.g., Nine Columns at Once
- 9 Columns  $\Leftrightarrow$  9 Windows
- 9 Windows  $\Leftrightarrow$  9 Histograms
- Histograms are Distributive
  - $H_{A \cup B}[v] = H_A[v] + H_B[v]$
  - $H_n = H_c + (H_n - H_c)$

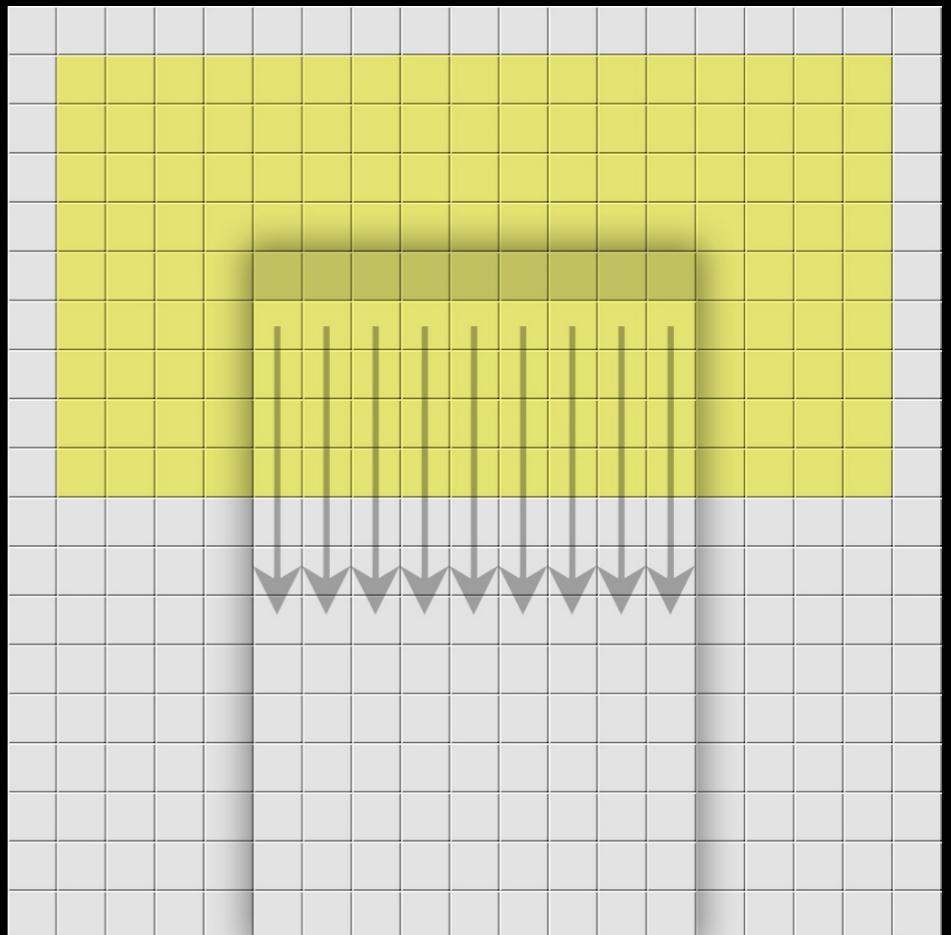
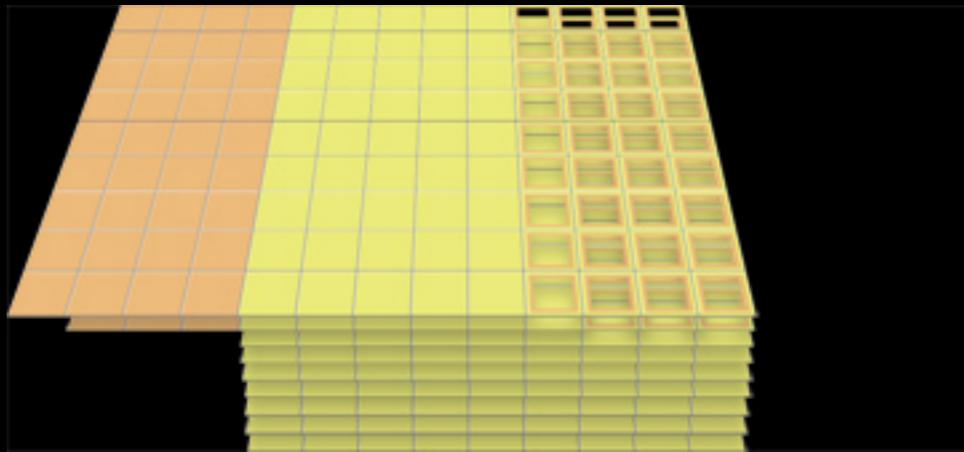




SIGGRAPH2006

# Parallel $O(r)$ Algorithm

- E.g., Nine Columns at Once
- 9 Columns  $\Leftrightarrow$  9 Windows
- 9 Windows  $\Leftrightarrow$  9 Histograms
- Histograms are Distributive
  - $H_{A \cup B}[v] = H_A[v] + H_B[v]$
  - $H_n = H_c + (H_n - H_c)$

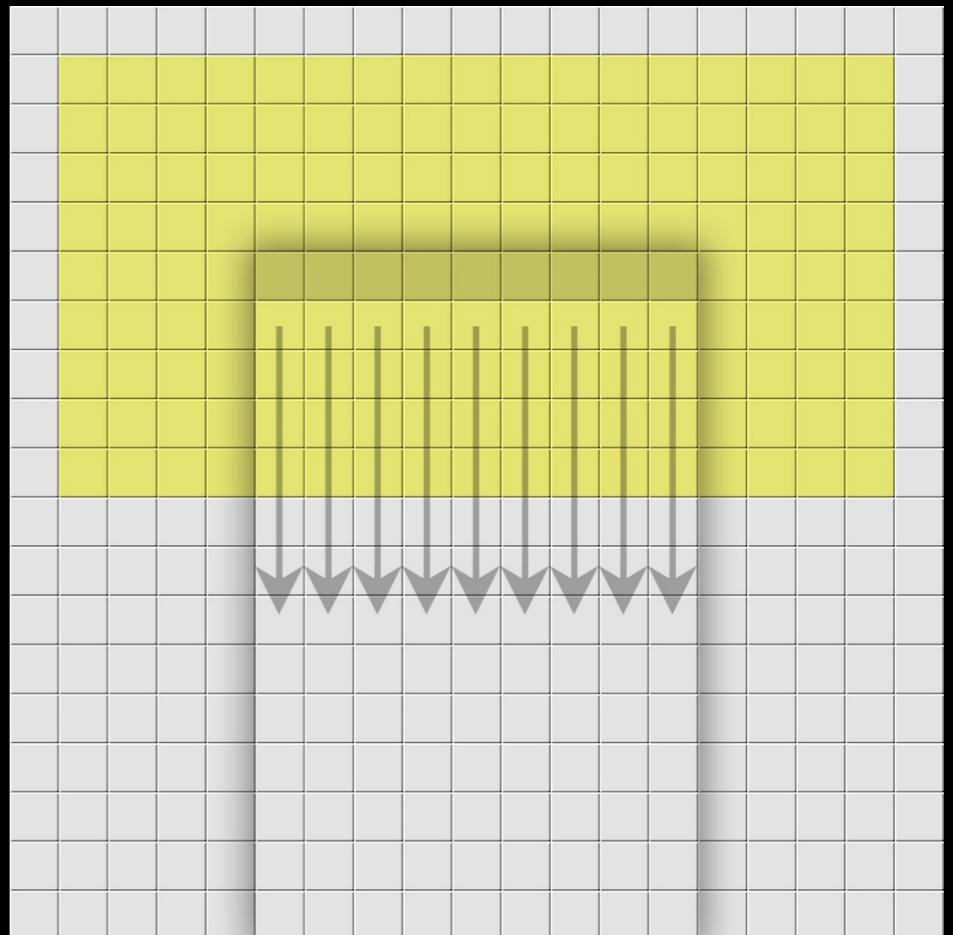
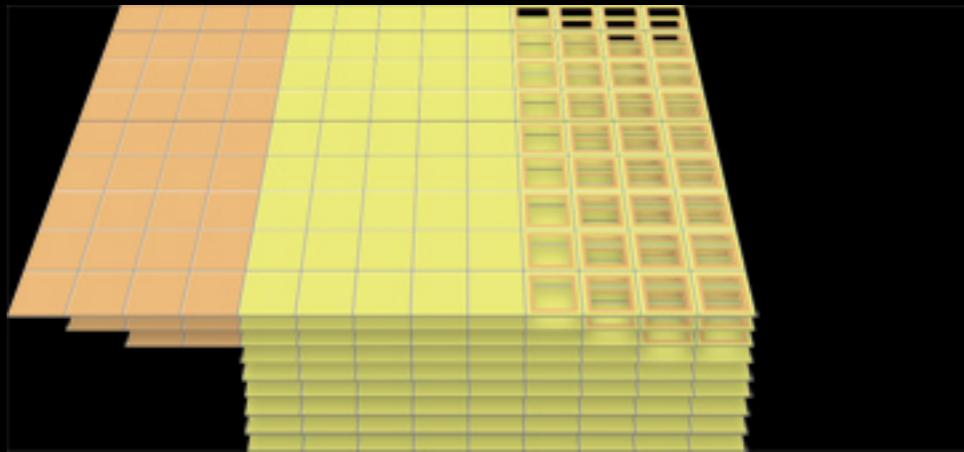




SIGGRAPH2006

# Parallel $O(r)$ Algorithm

- E.g., Nine Columns at Once
- 9 Columns  $\Leftrightarrow$  9 Windows
- 9 Windows  $\Leftrightarrow$  9 Histograms
- Histograms are Distributive
  - $H_{A \cup B}[v] = H_A[v] + H_B[v]$
  - $H_n = H_c + (H_n - H_c)$

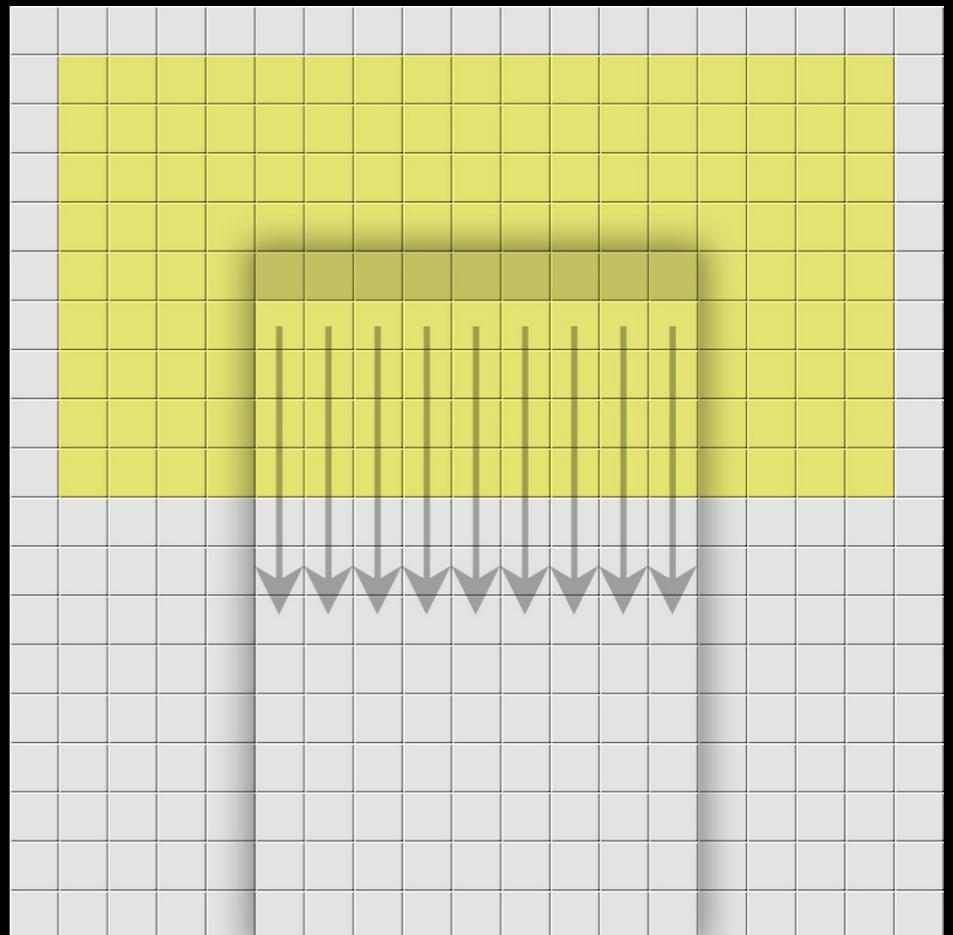
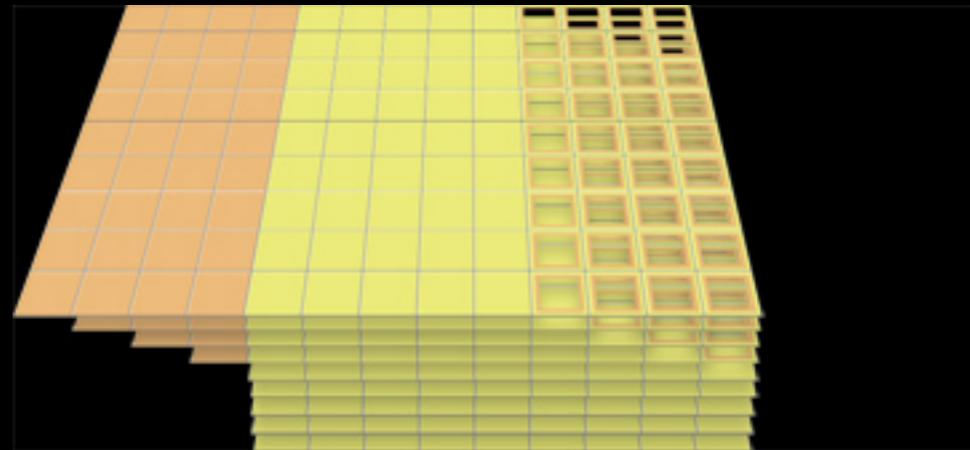




SIGGRAPH2006

# Parallel $O(r)$ Algorithm

- E.g., Nine Columns at Once
- 9 Columns  $\Leftrightarrow$  9 Windows
- 9 Windows  $\Leftrightarrow$  9 Histograms
- Histograms are **Distributive**
  - $H_{A \cup B}[v] = H_A[v] + H_B[v]$
  - $H_n = H_c + (H_n - H_c)$

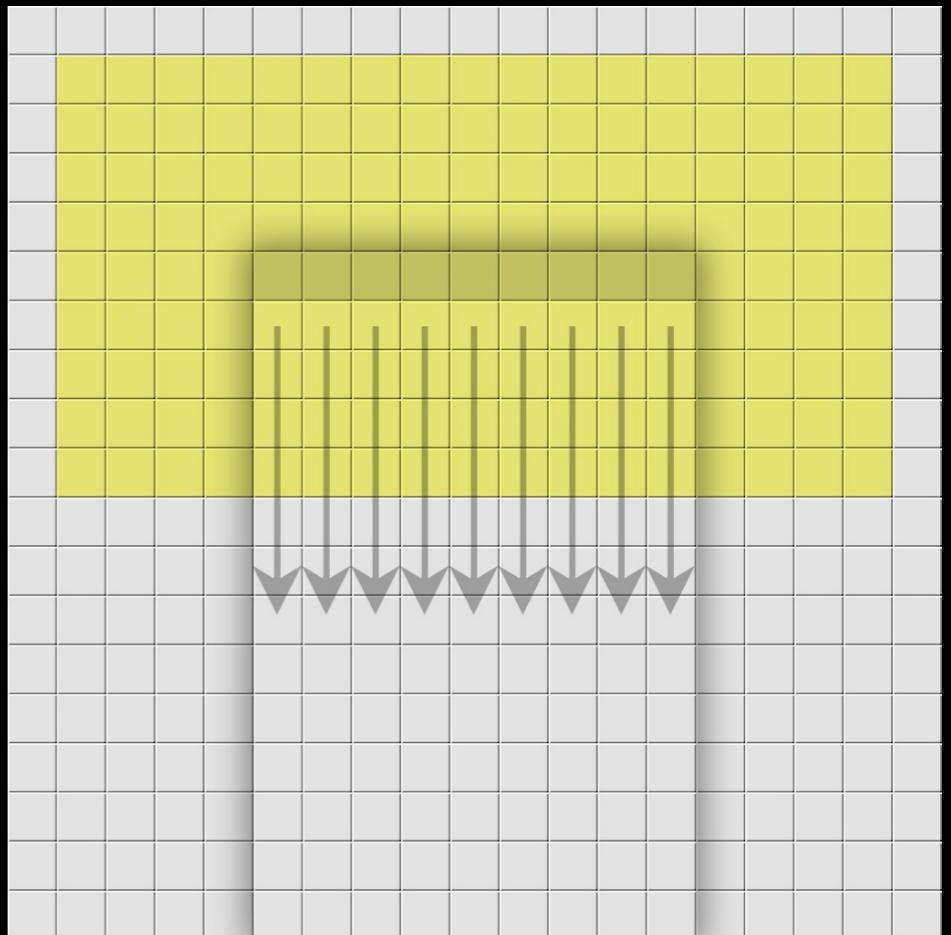
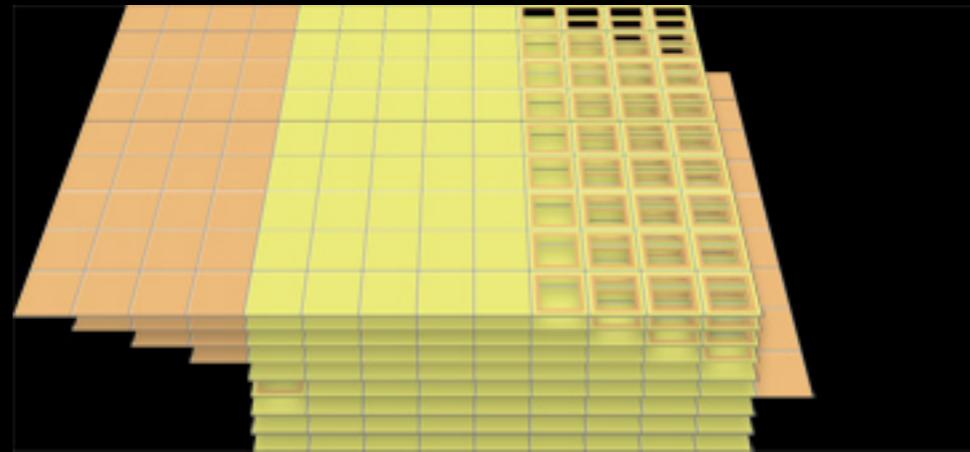




SIGGRAPH2006

# Parallel $O(r)$ Algorithm

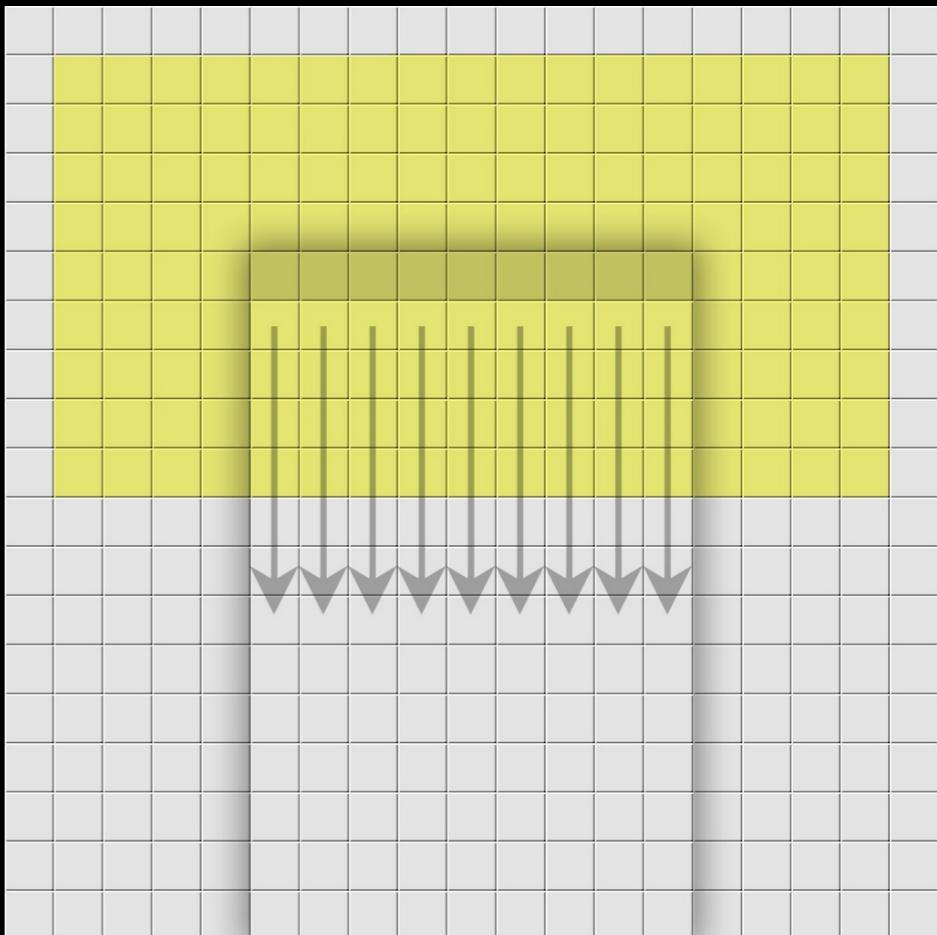
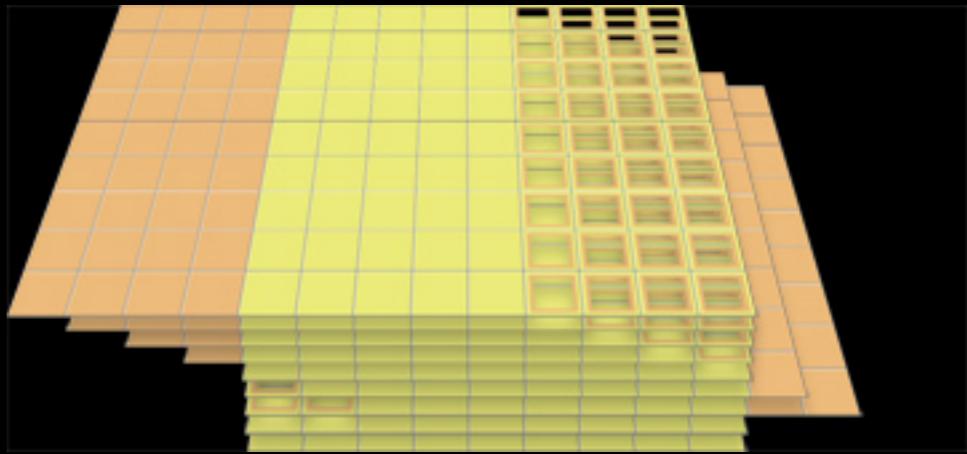
- E.g., Nine Columns at Once
- 9 Columns  $\Leftrightarrow$  9 Windows
- 9 Windows  $\Leftrightarrow$  9 Histograms
- Histograms are **Distributive**
  - $H_{A \cup B}[v] = H_A[v] + H_B[v]$
  - $H_n = H_c + (H_n - H_c)$



# Parallel $O(r)$ Algorithm



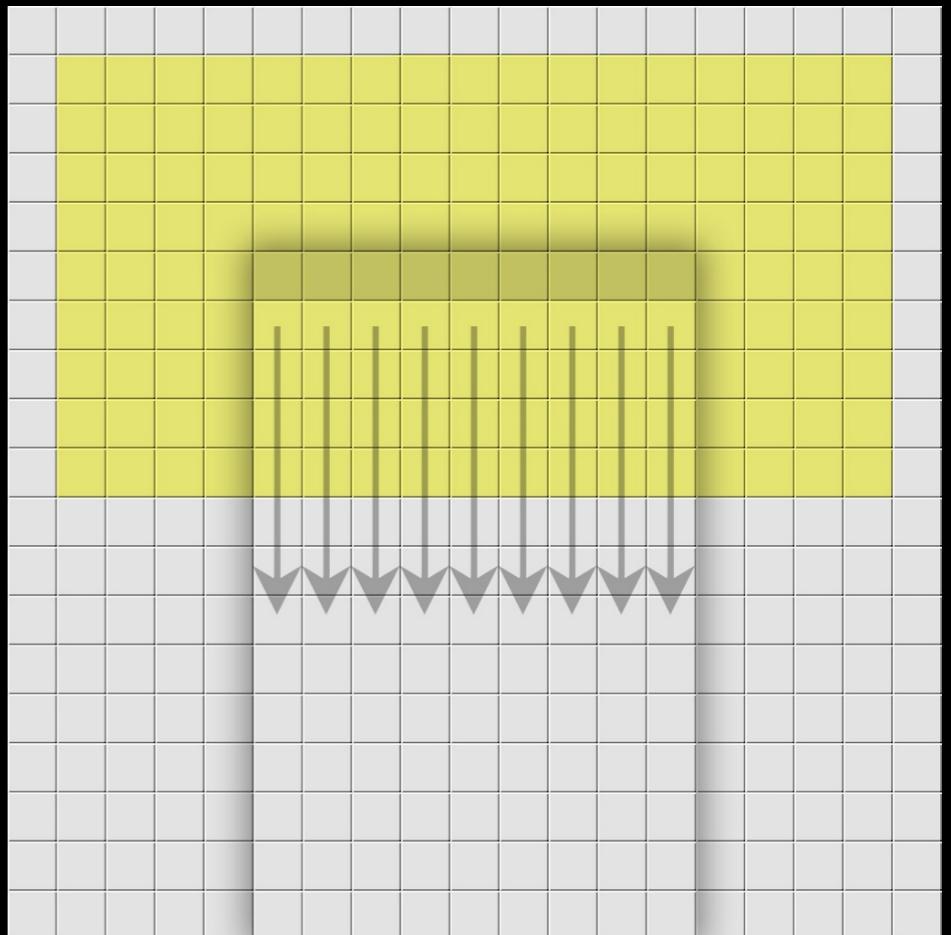
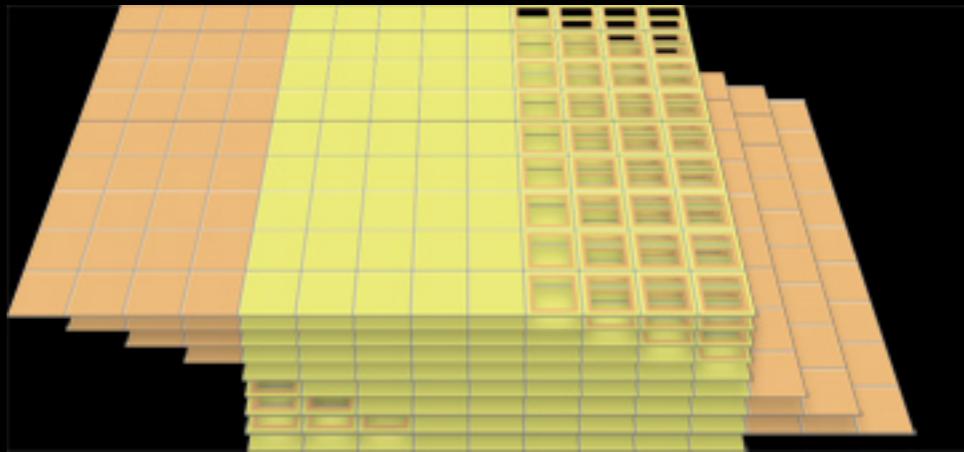
- E.g., Nine Columns at Once
- 9 Columns  $\Leftrightarrow$  9 Windows
- 9 Windows  $\Leftrightarrow$  9 Histograms
- Histograms are **Distributive**
  - $H_{A \cup B}[v] = H_A[v] + H_B[v]$
  - $H_n = H_c + (H_n - H_c)$



# Parallel $O(r)$ Algorithm



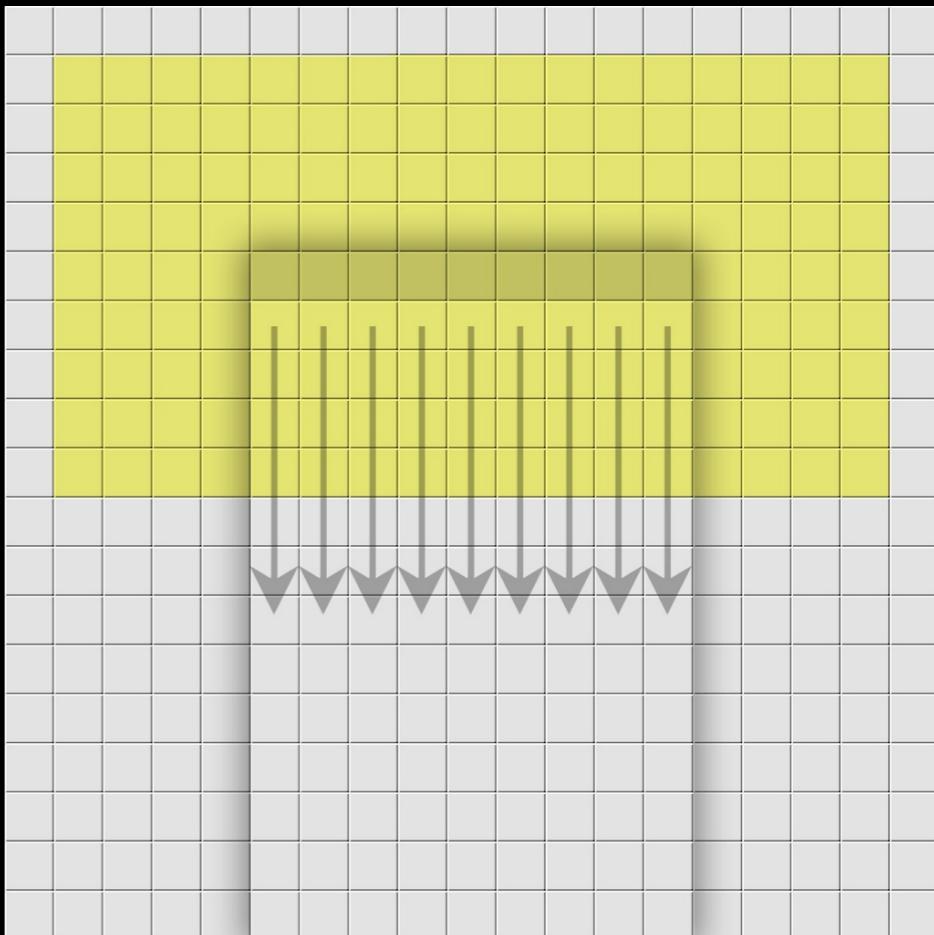
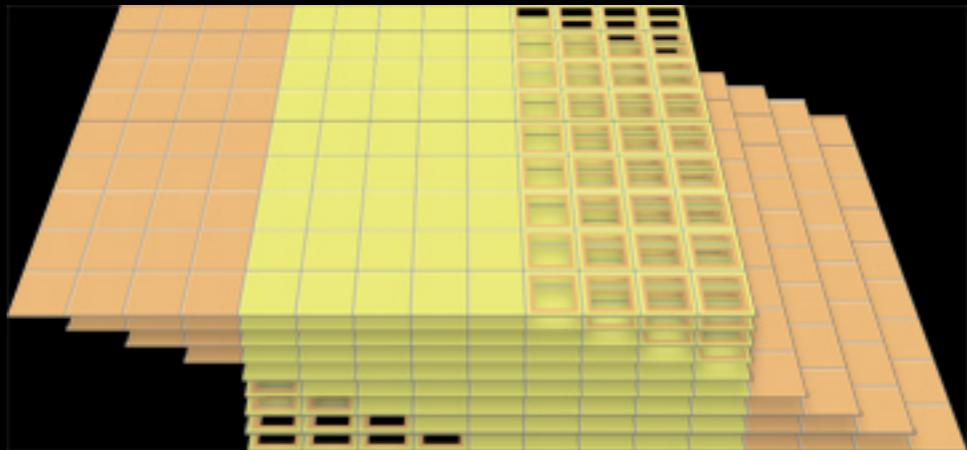
- E.g., Nine Columns at Once
- 9 Columns  $\Leftrightarrow$  9 Windows
- 9 Windows  $\Leftrightarrow$  9 Histograms
- Histograms are **Distributive**
  - $H_{A \cup B}[v] = H_A[v] + H_B[v]$
  - $H_n = H_c + (H_n - H_c)$



# Parallel $O(r)$ Algorithm

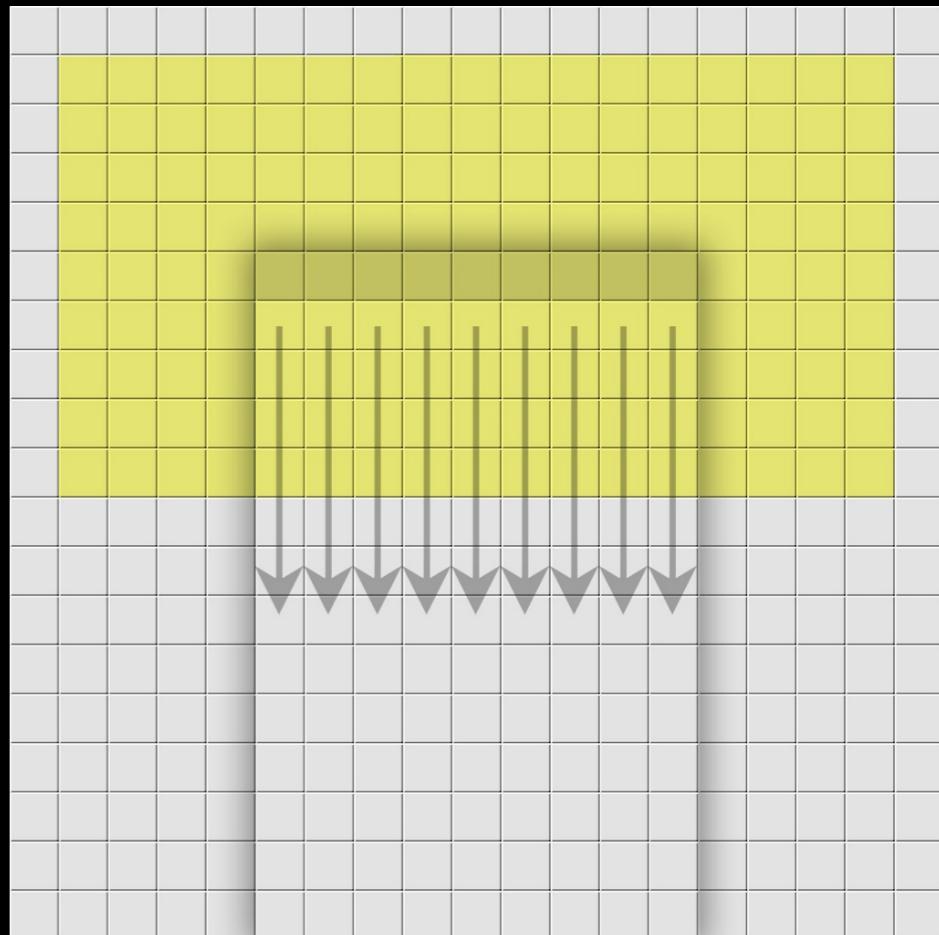
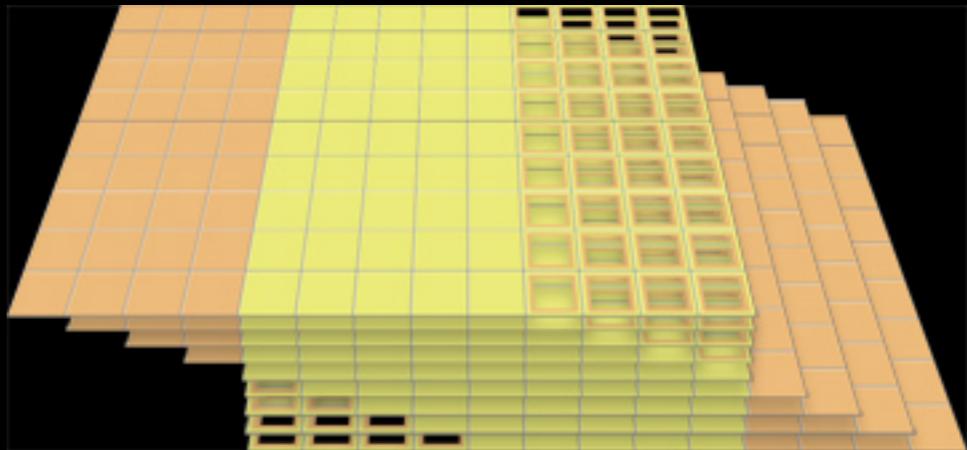


- E.g., Nine Columns at Once
- 9 Columns  $\Leftrightarrow$  9 Windows
- 9 Windows  $\Leftrightarrow$  9 Histograms
- Histograms are **Distributive**
  - $H_{A \cup B}[v] = H_A[v] + H_B[v]$
  - $H_n = H_c + (H_n - H_c)$



# Parallel $O(r)$ Algorithm

- E.g., Nine Columns at Once
- 9 Columns  $\Leftrightarrow$  9 Windows
- 9 Windows  $\Leftrightarrow$  9 Histograms
- Histograms are **Distributive**
  - $H_{A \cup B}[v] = H_A[v] + H_B[v]$
  - $H_n = H_c + (H_n - H_c)$
- Approaches ninefold speedup



# Adaptive $O(r^{1/2})$ Algorithm



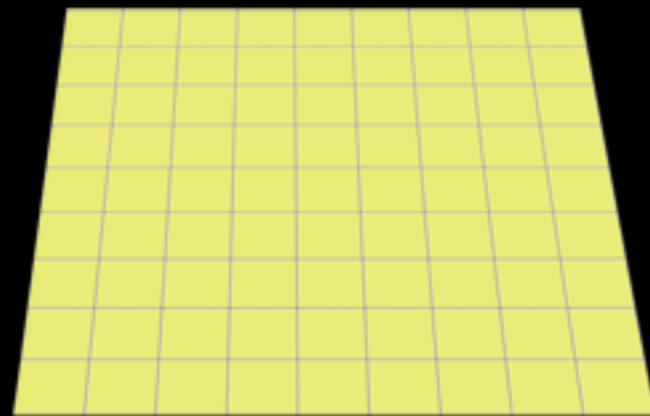
SIGGRAPH2006



SIGGRAPH2006

# Adaptive $O(r^{1/2})$ Algorithm

- For Fixed Radius  $r$ ,  $N$  is Variable

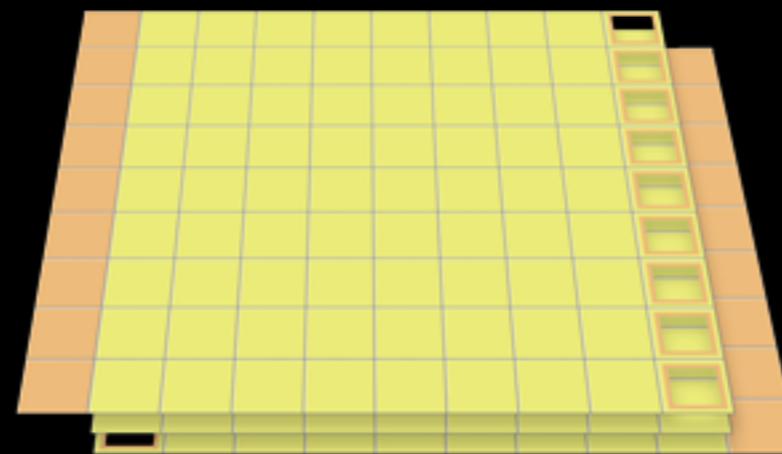




SIGGRAPH2006

# Adaptive $O(r^{1/2})$ Algorithm

- For Fixed Radius  $r$ ,  $N$  is Variable

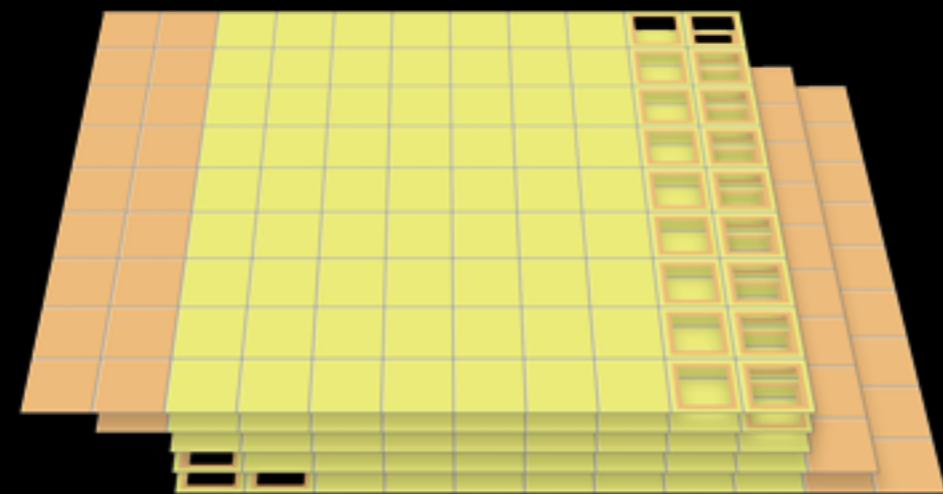




SIGGRAPH2006

# Adaptive $O(r^{1/2})$ Algorithm

- For Fixed Radius  $r$ ,  $N$  is Variable

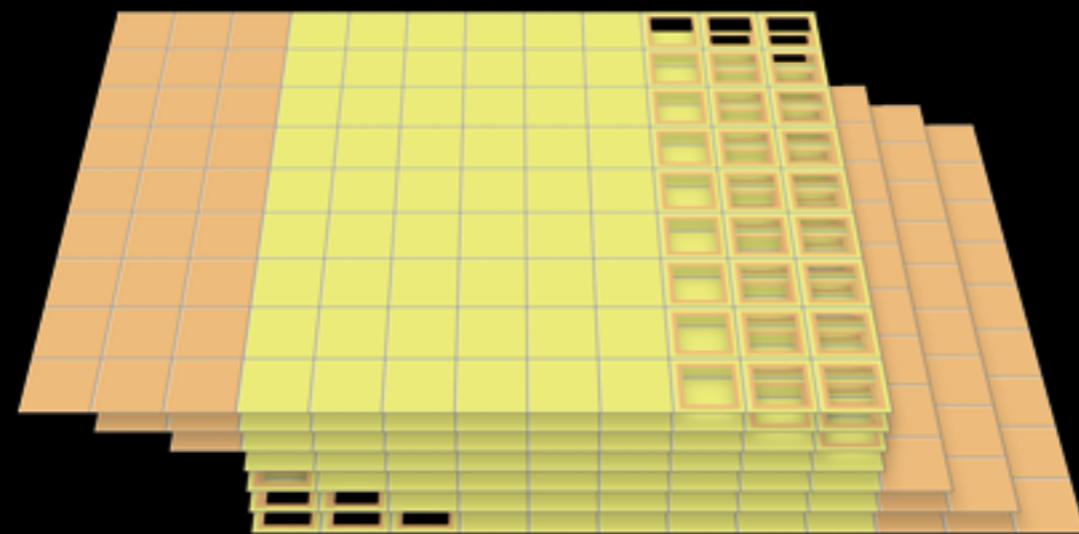




SIGGRAPH2006

# Adaptive $O(r^{1/2})$ Algorithm

- For Fixed Radius  $r$ ,  $N$  is Variable

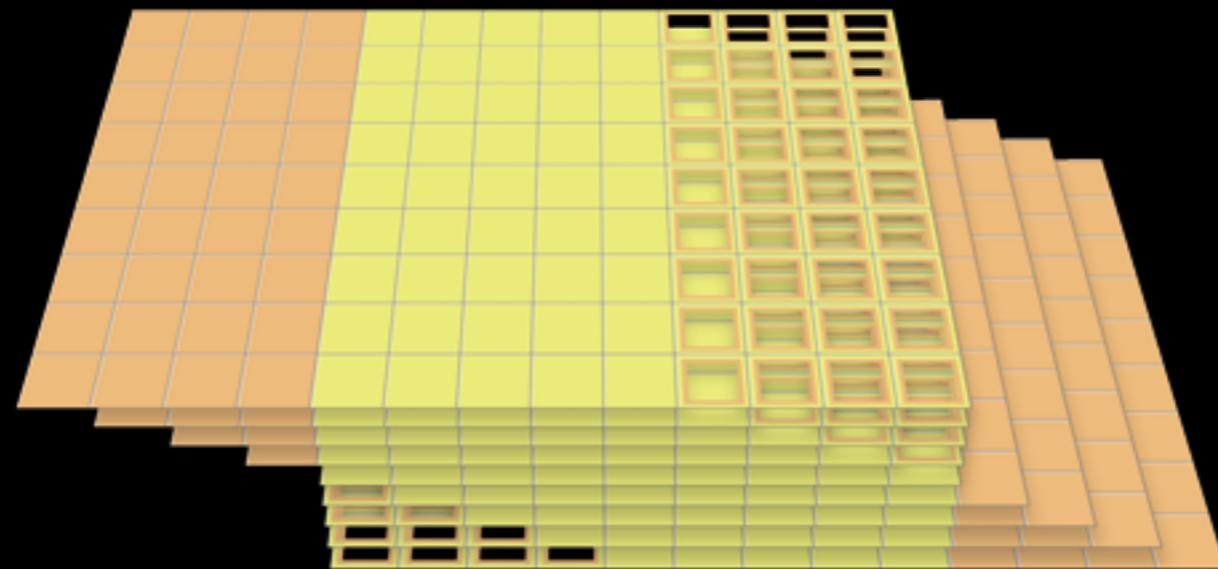




SIGGRAPH2006

# Adaptive $O(r^{1/2})$ Algorithm

- For Fixed Radius  $r$ ,  $N$  is Variable

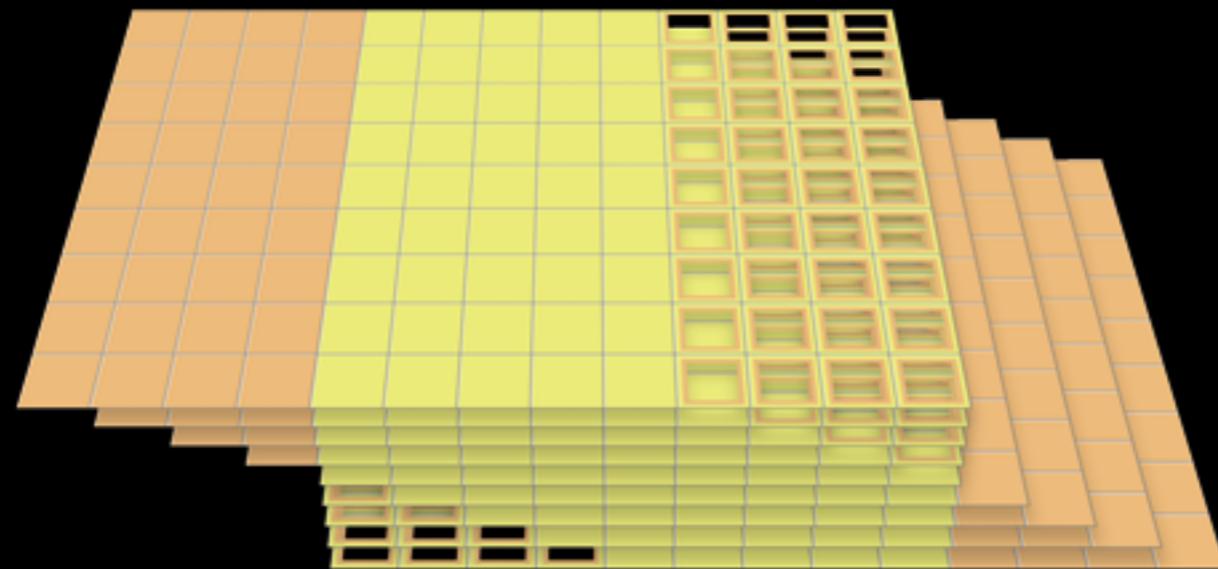




SIGGRAPH2006

# Adaptive $O(r^{1/2})$ Algorithm

- For Fixed Radius  $r$ ,  $N$  is Variable
- For Fixed  $N$ , Radius  $r$  is Variable

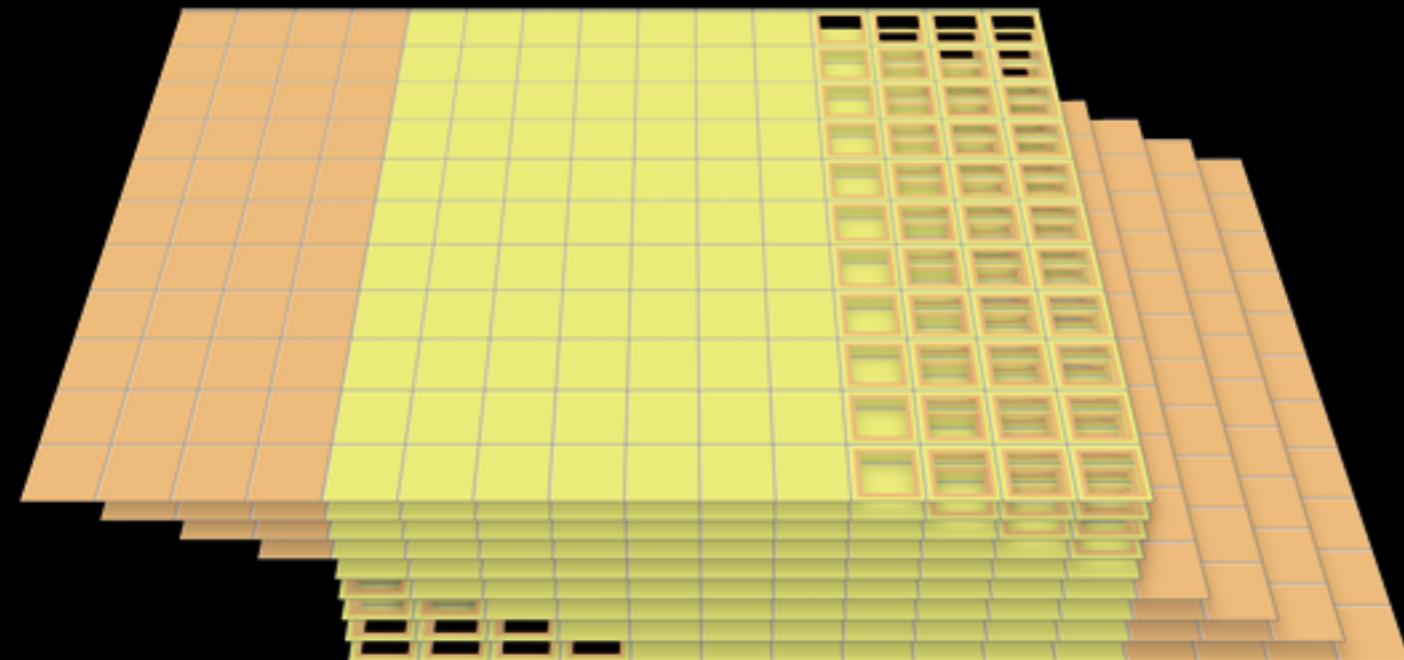




SIGGRAPH2006

# Adaptive $O(r^{1/2})$ Algorithm

- For Fixed Radius  $r$ ,  $N$  is Variable
- For Fixed  $N$ , Radius  $r$  is Variable

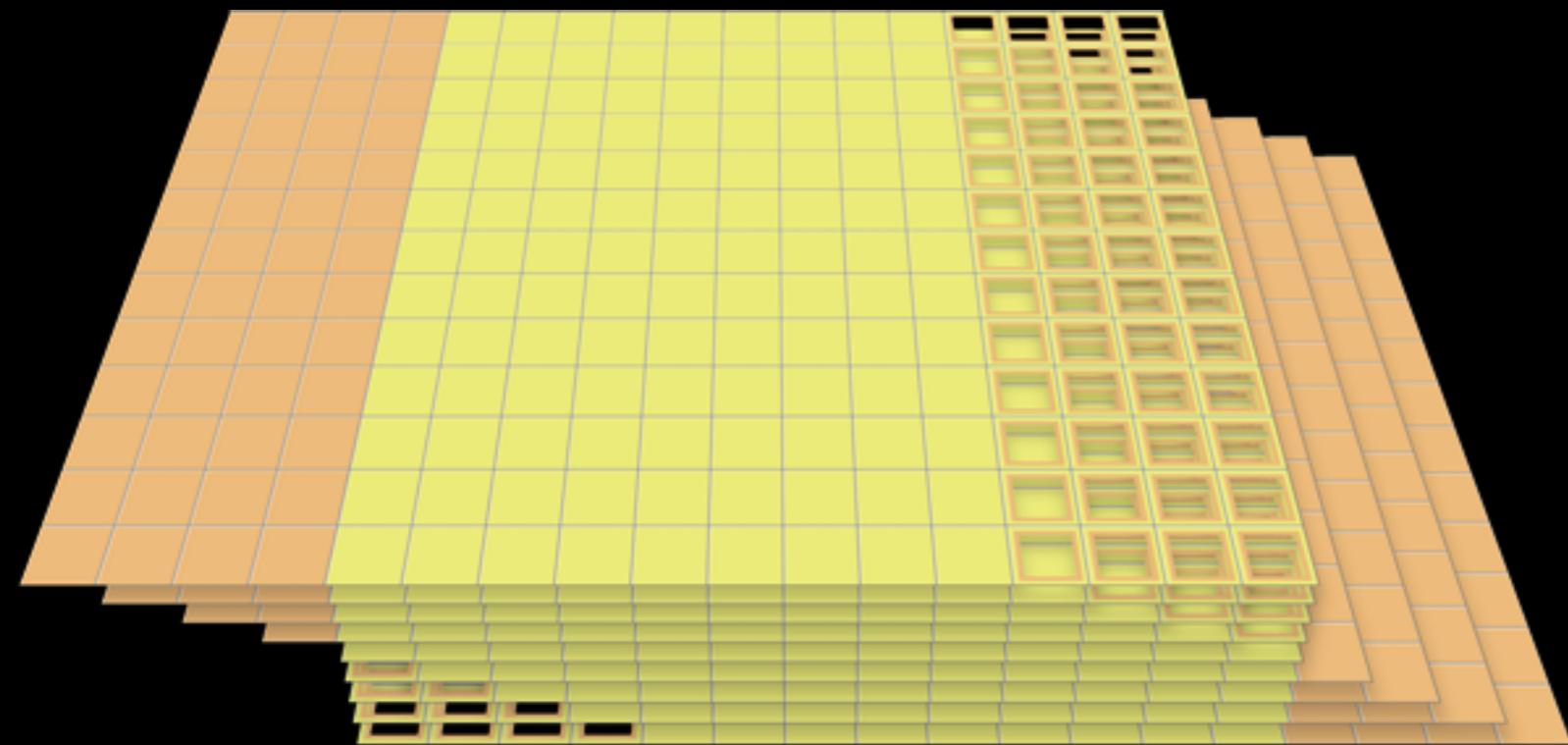




SIGGRAPH2006

# Adaptive $O(r^{1/2})$ Algorithm

- For Fixed Radius  $r$ ,  $N$  is Variable
- For Fixed  $N$ , Radius  $r$  is Variable

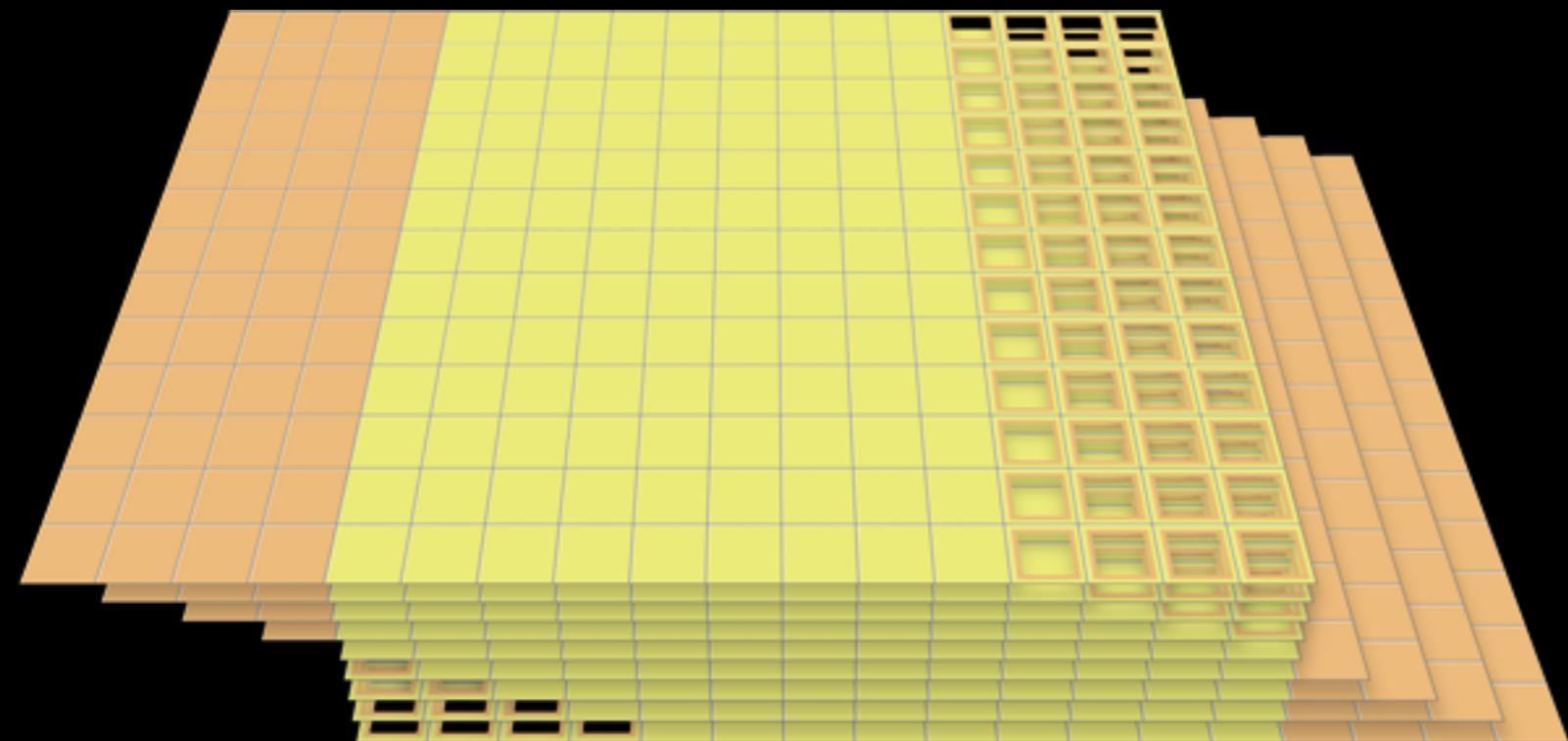




SIGGRAPH2006

# Adaptive $O(r^{1/2})$ Algorithm

- For Fixed Radius  $r$ ,  $N$  is Variable
- For Fixed  $N$ , Radius  $r$  is Variable
- $N \approx r^{1/2}$  yields  $O(r^{1/2})$  Complexity





SIGGRAPH2006

# The $O(\log r)$ Algorithm

---



SIGGRAPH2006

# The $O(\log r)$ Algorithm

---

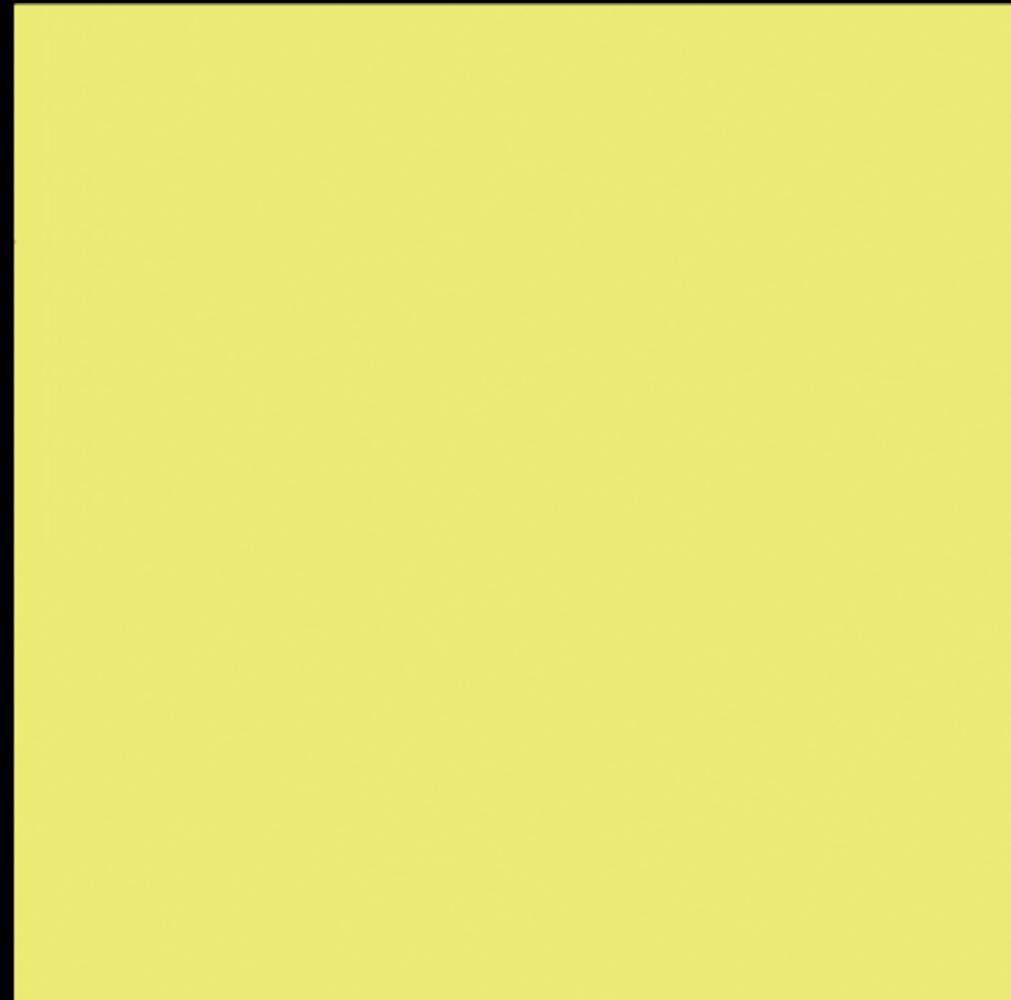
- Add Histogram *Tiers* with Radius



SIGGRAPH2006

# The $O(\log r)$ Algorithm

- Add Histogram *Tiers* with Radius
  - $H_n = H_c$

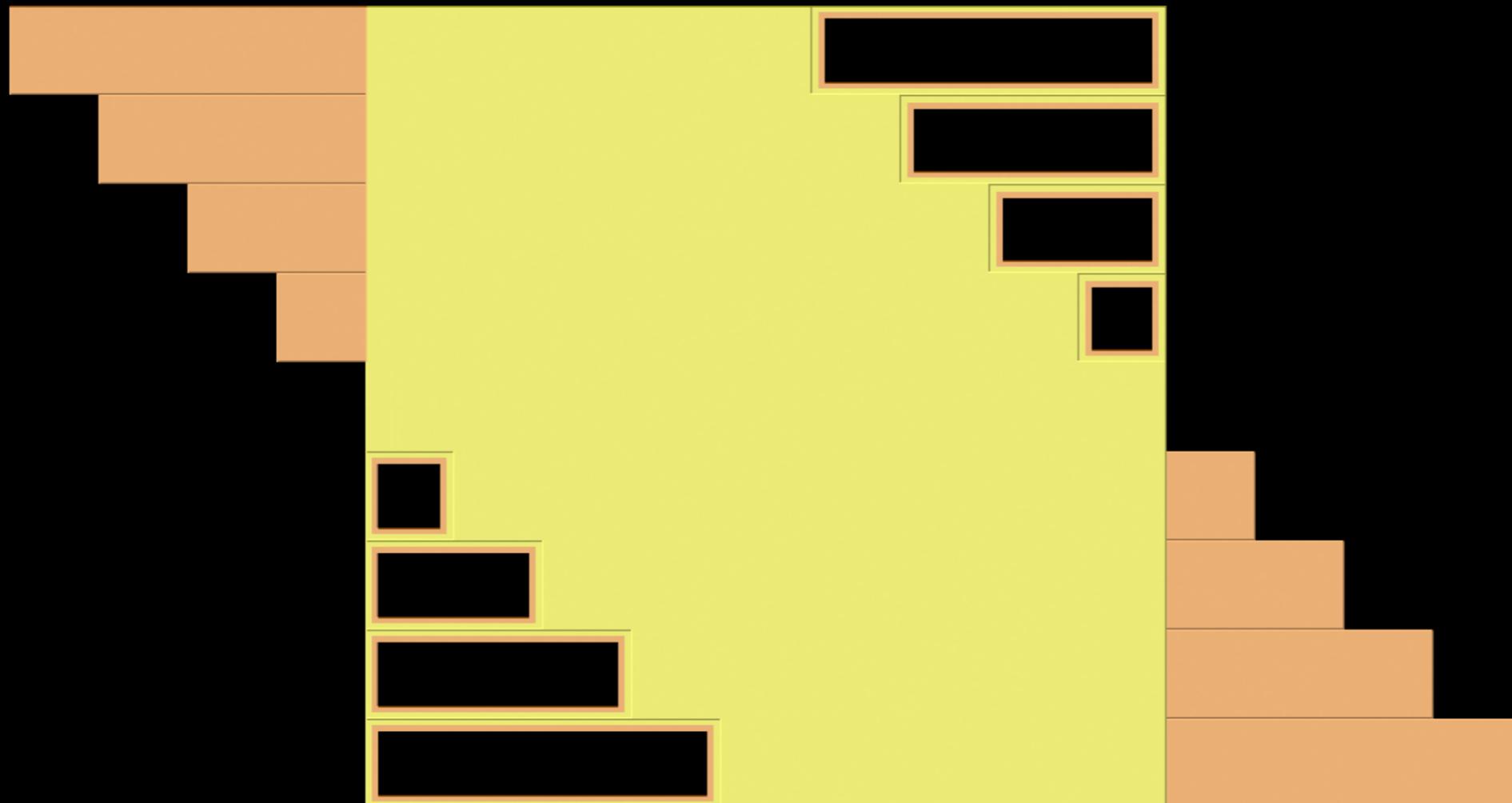




SIGGRAPH2006

# The $O(\log r)$ Algorithm

- Add Histogram *Tiers* with Radius
  - $H_n = H_c + (H_d - H_c)$

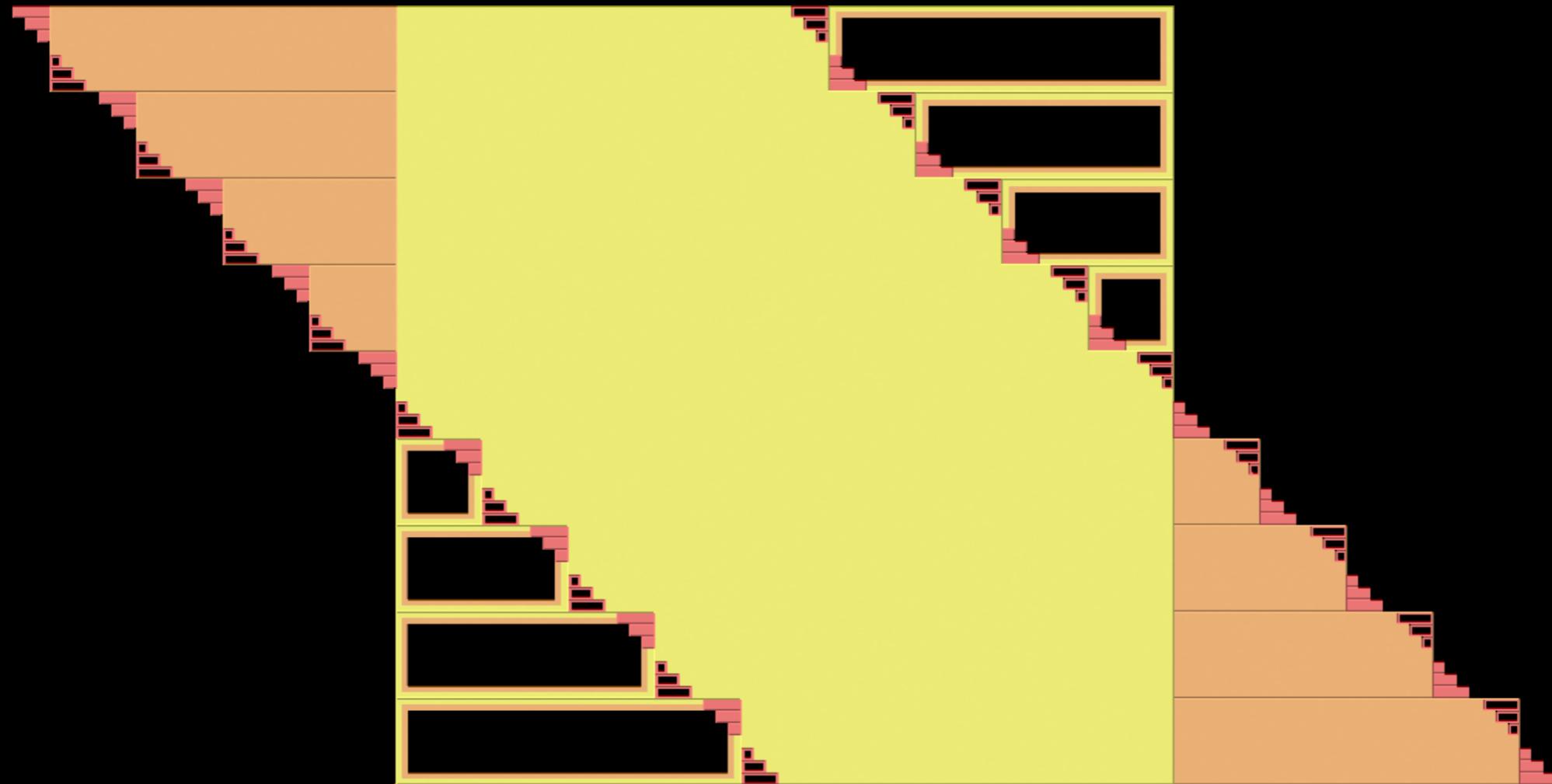




SIGGRAPH2006

# The $O(\log r)$ Algorithm

- Add Histogram *Tiers* with Radius
  - $H_n = H_c + (H_d - H_c) + (H_n - H_d)$

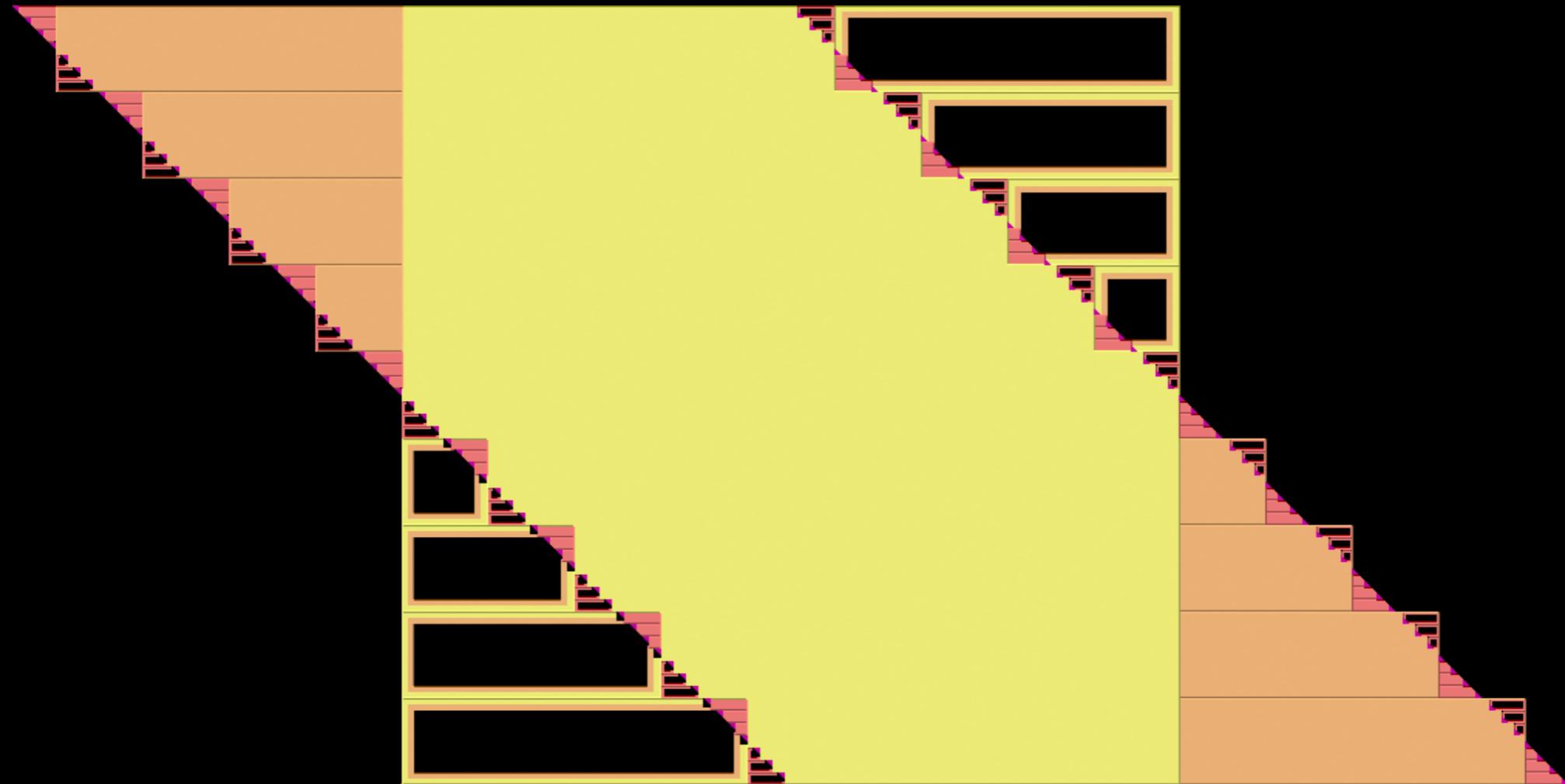




SIGGRAPH2006

# The $O(\log r)$ Algorithm

- Add Histogram *Tiers* with Radius
  - $H_n = H_c + (H_d - H_c) + (H_n - H_d) + \dots$

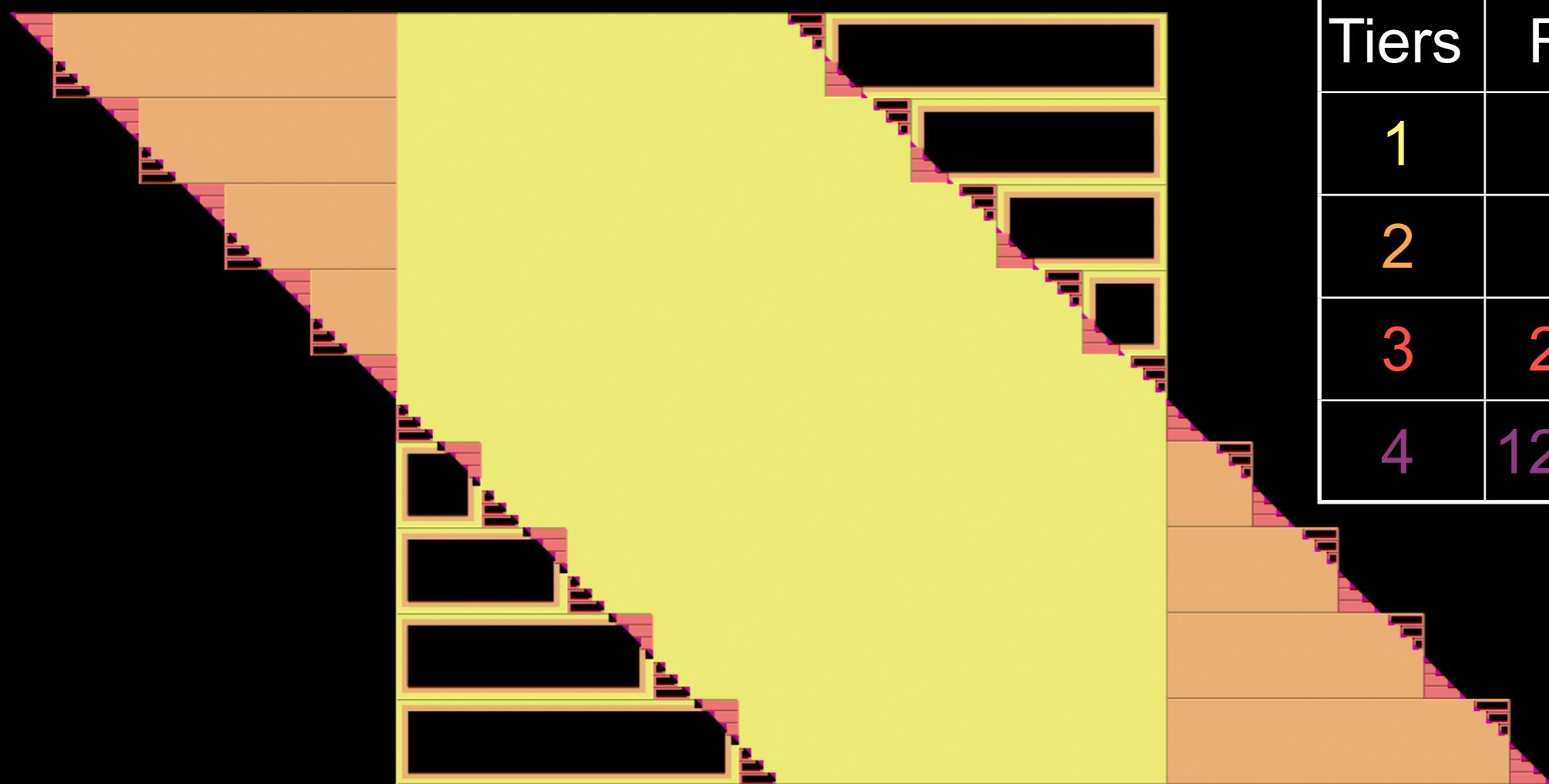




SIGGRAPH2006

# The $O(\log r)$ Algorithm

- Add Histogram *Tiers* with Radius
  - $H_n = H_c + (H_d - H_c) + (H_n - H_d) + \dots$
- Constant Radix yields  $O(\log r)$  Algorithm

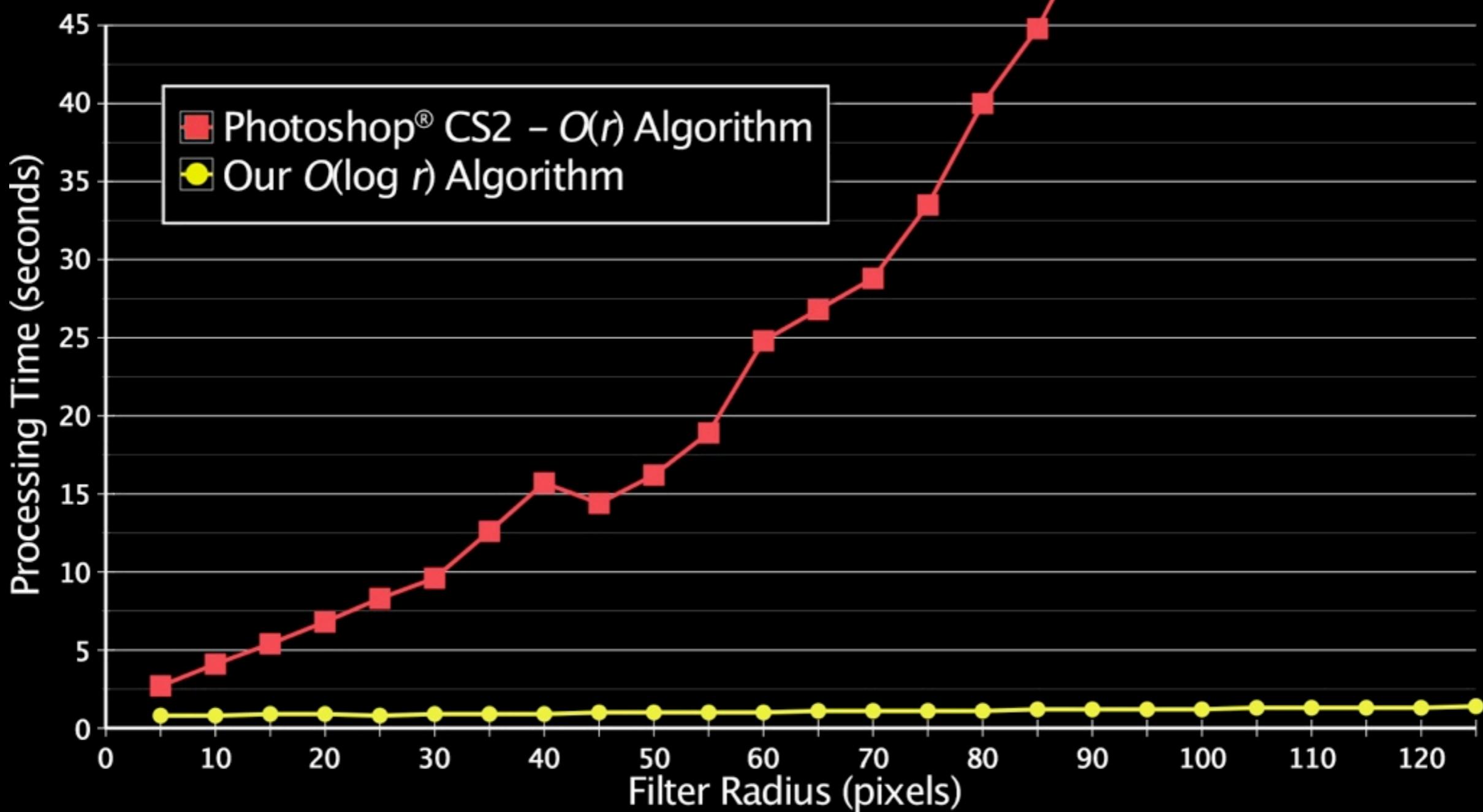


# 8-Bit Median - Performance



SIGGRAPH2006

PowerMac G5 2.5GHz – Single Processor – 8-Megapixel RGB Image



# 8-Bit Median - Performance



SIGGRAPH2006

Demo

# 8-Bit Algorithm $\Rightarrow$ 16-Bit



SIGGRAPH2006

# 8-Bit Algorithm $\Rightarrow$ 16-Bit



SIGGRAPH2006

- What's a Few Extra Bits?



SIGGRAPH2006

# 8-Bit Algorithm $\Rightarrow$ 16-Bit

---

- What's a Few Extra Bits?
- Histogram Size: 256  $\Rightarrow$  65536 Elements!



SIGGRAPH2006

# 8-Bit Algorithm $\Rightarrow$ 16-Bit

---

- What's a Few Extra Bits?
- Histogram Size: 256  $\Rightarrow$  65536 Elements!
- But... Histogram becomes Sparse



SIGGRAPH2006

# 8-Bit Algorithm $\Rightarrow$ 16-Bit

---

- What's a Few Extra Bits?
- Histogram Size: 256  $\Rightarrow$  65536 Elements!
- But... Histogram becomes Sparse
  - Many Elements will be 0 or 1



SIGGRAPH2006

# 8-Bit Algorithm $\Rightarrow$ 16-Bit

---

- What's a Few Extra Bits?
- Histogram Size: 256  $\Rightarrow$  65536 Elements!
- But... Histogram becomes Sparse
  - Many Elements will be 0 or 1
  - Still Need to Handle All Cases



SIGGRAPH2006

# 8-Bit Algorithm $\Rightarrow$ 16-Bit

---

- What's a Few Extra Bits?
- Histogram Size: 256  $\Rightarrow$  65536 Elements!
- But... Histogram becomes Sparse
  - Many Elements will be 0 or 1
  - Still Need to Handle All Cases
- Solution?

# The Ordinal Transform



SIGGRAPH2006

# The Ordinal Transform



SIGGRAPH2006

- Make Each Source Value Unique

# The Ordinal Transform



SIGGRAPH2006

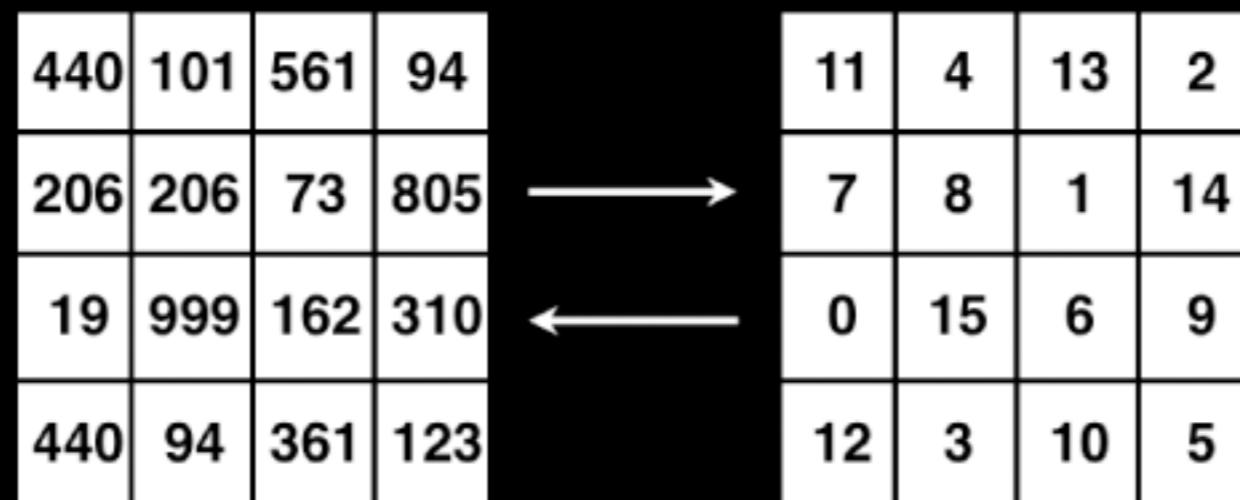
- Make Each Source Value Unique
  - Enables Binary Histograms, 16x Smaller



SIGGRAPH2006

# The Ordinal Transform

- Make Each Source Value Unique
  - Enables Binary Histograms, 16x Smaller
- Technique: Cardinal  $\Leftrightarrow$  Ordinal Mapping

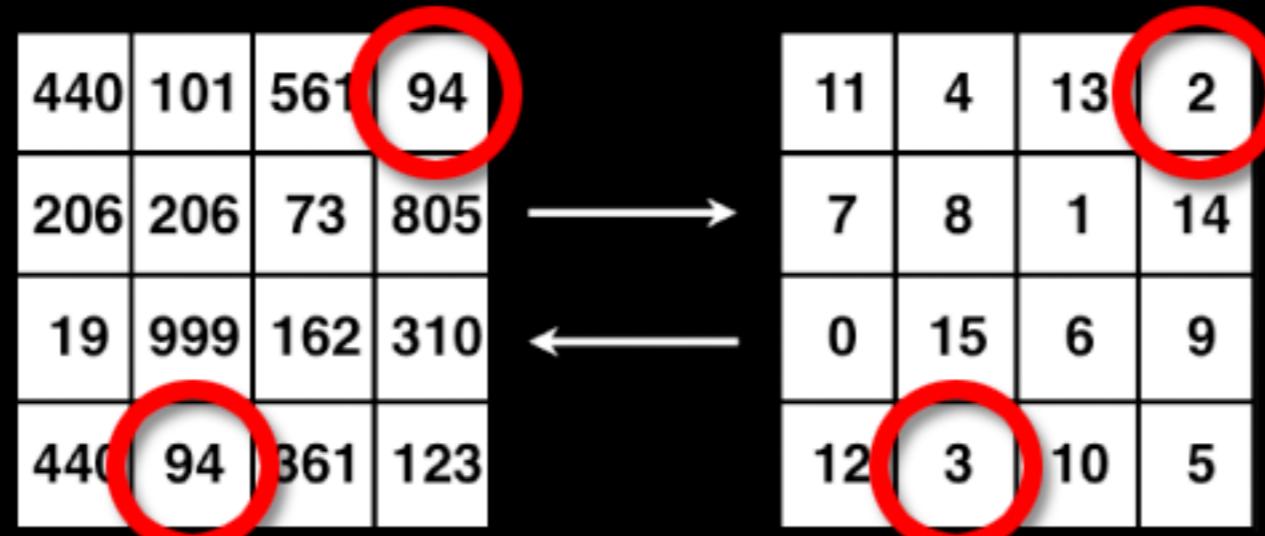




SIGGRAPH2006

# The Ordinal Transform

- Make Each Source Value Unique
  - Enables Binary Histograms, 16x Smaller
- Technique: Cardinal  $\Leftrightarrow$  Ordinal Mapping

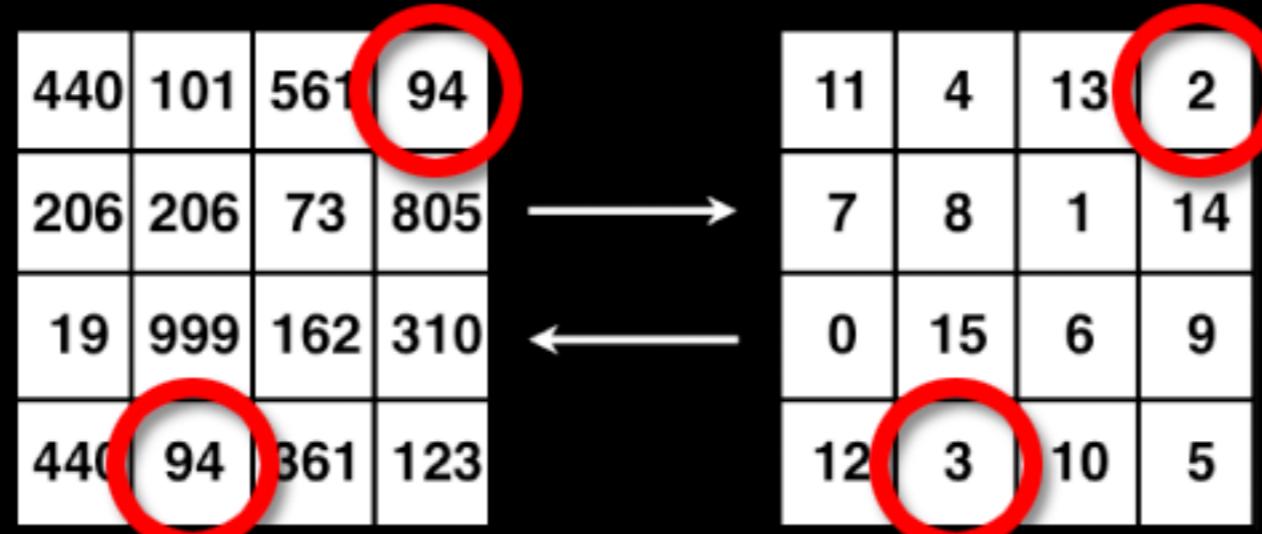


# The Ordinal Transform



SIGGRAPH2006

- Make Each Source Value Unique
  - Enables Binary Histograms, 16x Smaller
- Technique: Cardinal  $\Leftrightarrow$  Ordinal Mapping
- Median is Invariant under Ordinal Transform!

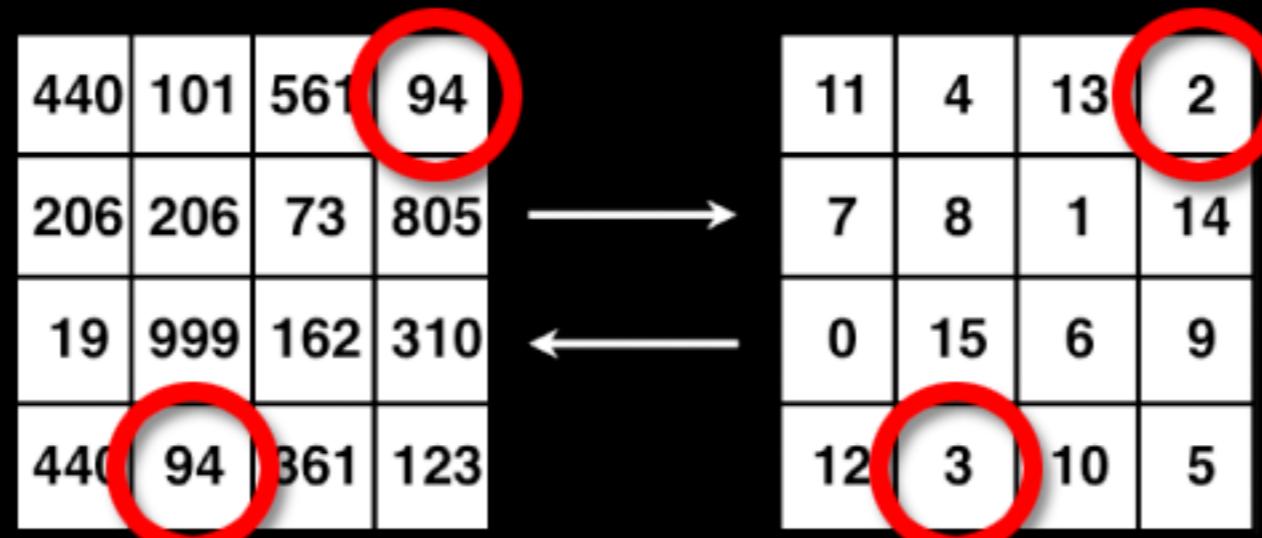


# The Ordinal Transform



SIGGRAPH2006

- Make Each Source Value Unique
  - Enables Binary Histograms, 16x Smaller
- Technique: Cardinal  $\Leftrightarrow$  Ordinal Mapping
- Median is Invariant under Ordinal Transform!
- Apply Median to *Ordinal* Image



# The Compound Histogram



SIGGRAPH2006



SIGGRAPH2006

# The Compound Histogram

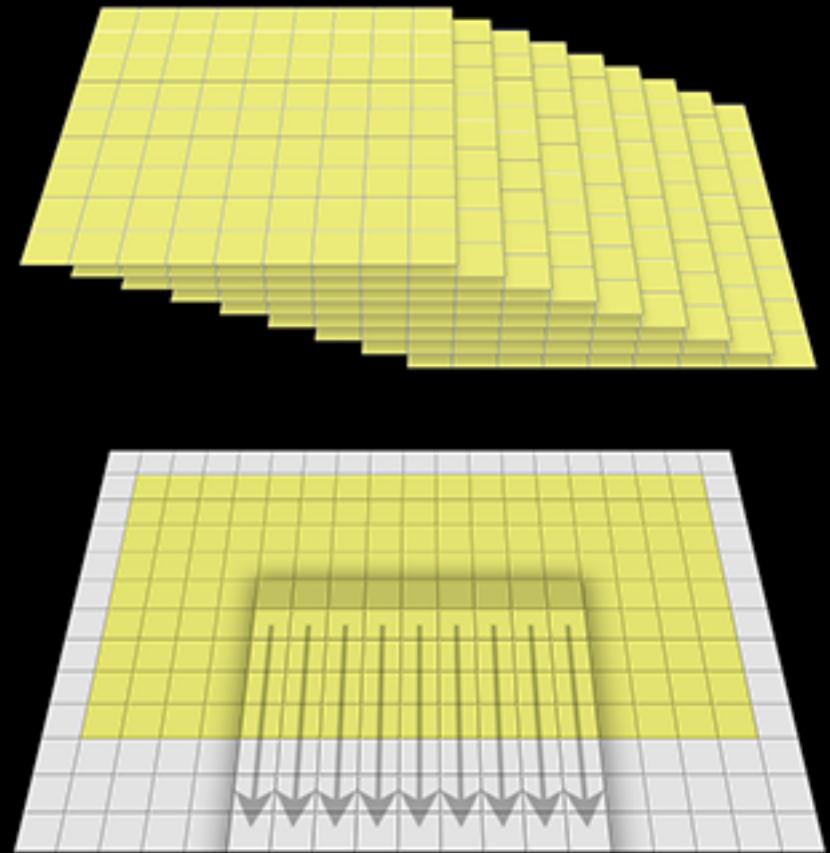
- Redundancy in Binary  $H_n$



SIGGRAPH2006

# The Compound Histogram

- Redundancy in Binary  $H_n$

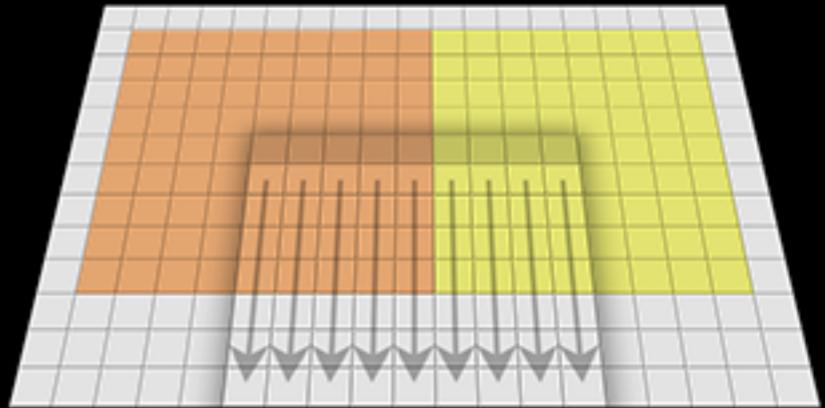
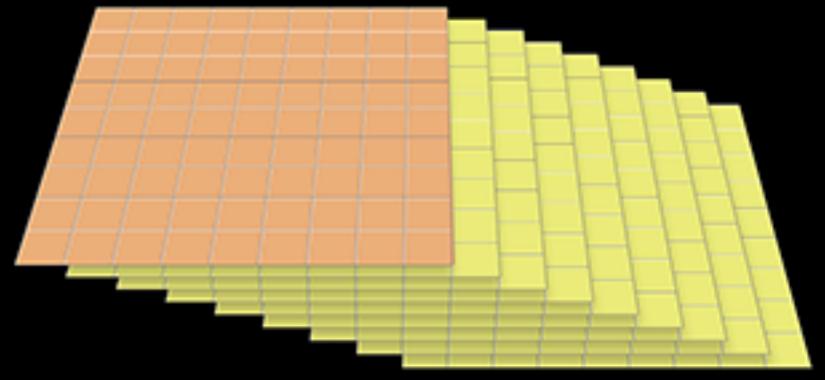




SIGGRAPH2006

# The Compound Histogram

- Redundancy in Binary  $H_n$



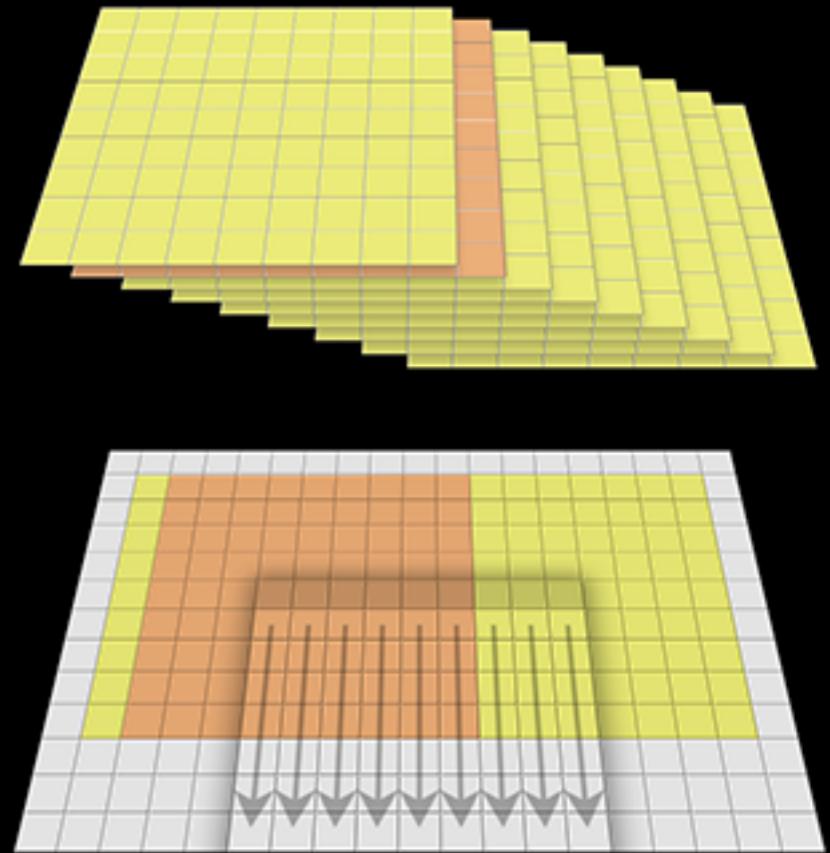
1  
0  
0  
0  
1  
1  
0  
1  
0



SIGGRAPH2006

# The Compound Histogram

- Redundancy in Binary  $H_n$



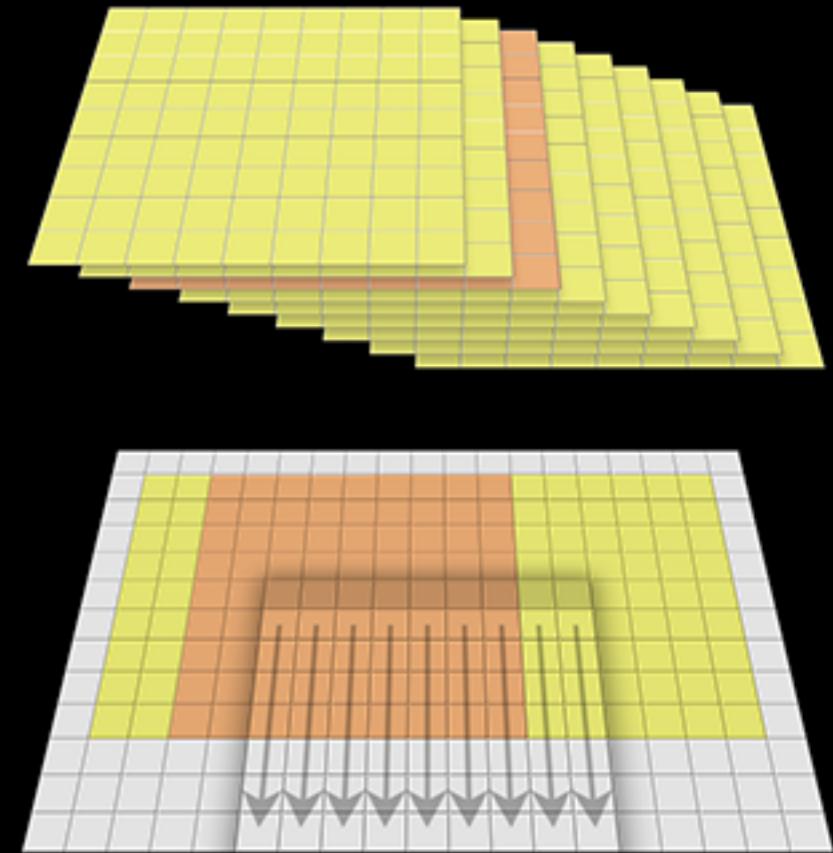
1	1
0	0
0	0
0	0
0	0
1	1
1	1
0	0
1	1
0	0



SIGGRAPH2006

# The Compound Histogram

- Redundancy in Binary  $H_n$



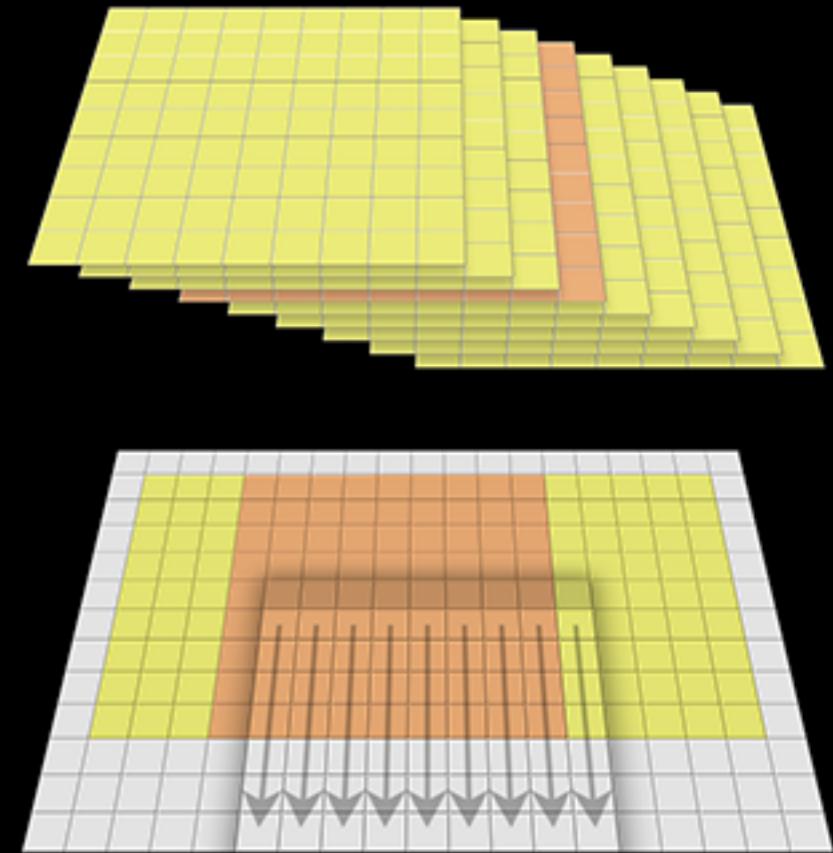
1	1	1
0	0	0
0	0	0
0	0	0
0	0	0
1	1	1
1	1	1
0	0	0
1	1	1
0	0	0



SIGGRAPH2006

# The Compound Histogram

- Redundancy in Binary  $H_n$



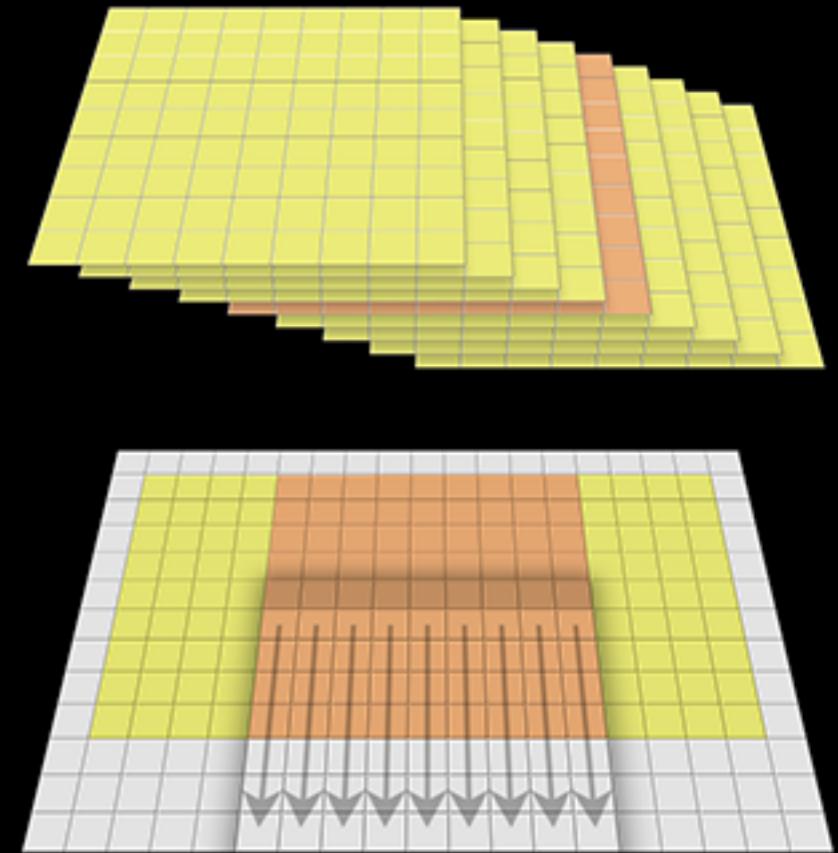
1	1	1	1
0	0	0	0
0	0	0	0
0	0	0	1
0	0	0	0
1	1	1	1
1	1	1	1
0	0	0	0
1	1	1	0
0	0	0	0



SIGGRAPH2006

# The Compound Histogram

- Redundancy in Binary  $H_n$



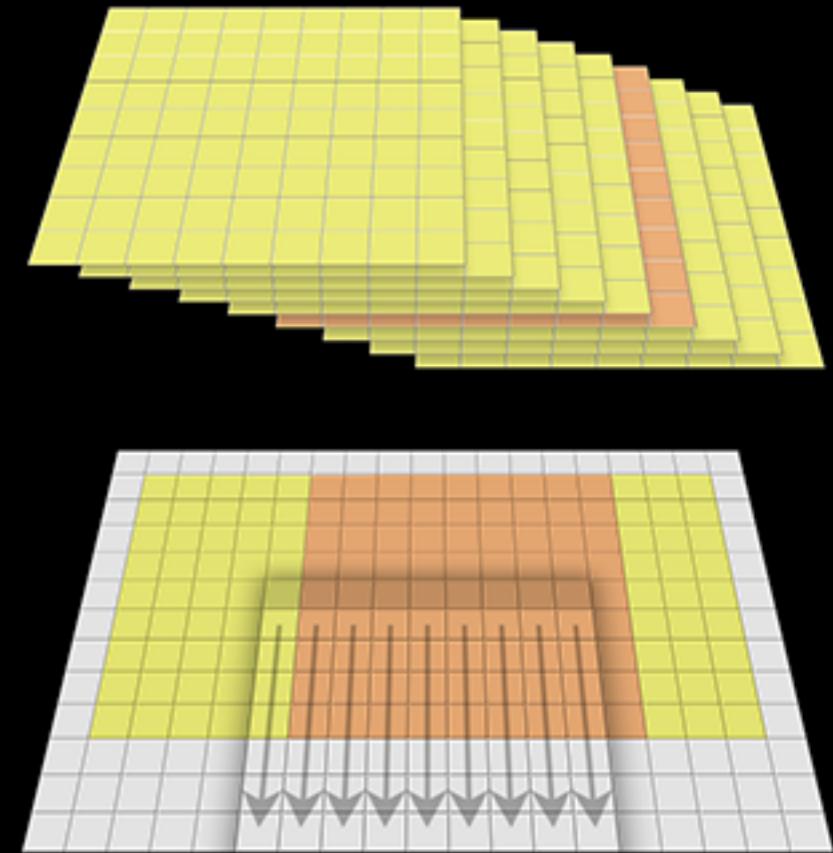
1	1	1	1	1
0	0	0	0	1
0	0	0	0	0
0	0	0	1	1
0	0	0	0	0
1	1	1	1	1
1	1	1	1	1
0	0	0	0	0
1	1	1	0	0
0	0	0	0	0



SIGGRAPH2006

# The Compound Histogram

- Redundancy in Binary  $H_n$



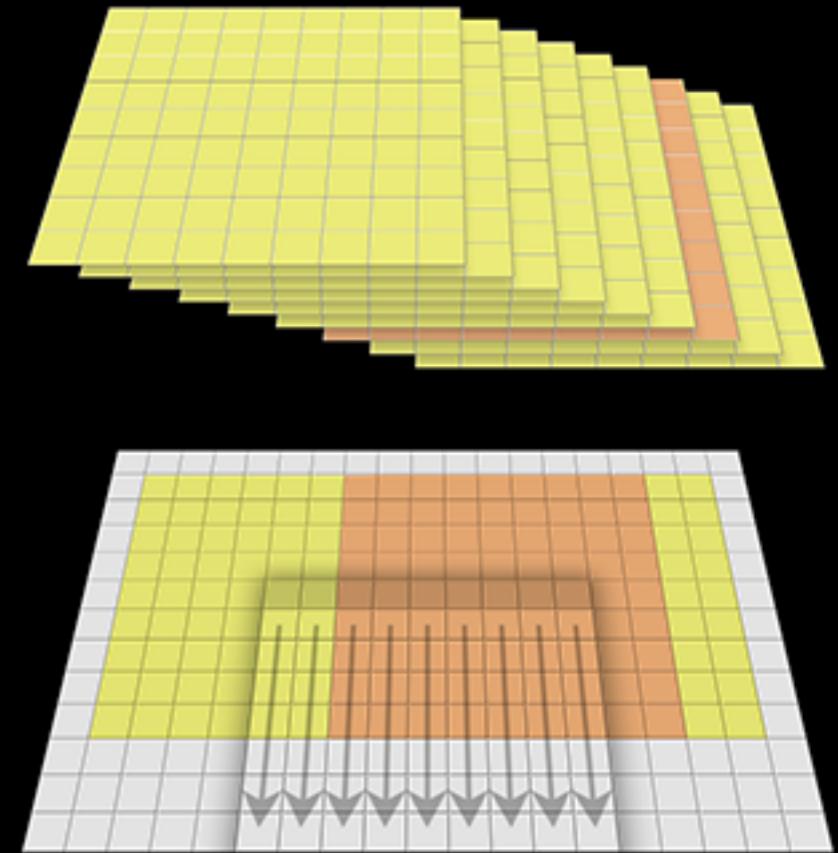
1	1	1	1	1	1
0	0	0	0	1	1
0	0	0	0	0	0
0	0	0	1	1	1
0	0	0	0	0	0
1	1	1	1	1	0
1	1	1	1	1	1
0	0	0	0	0	0
1	1	1	0	0	0
0	0	0	0	0	0



SIGGRAPH2006

# The Compound Histogram

- Redundancy in Binary  $H_n$



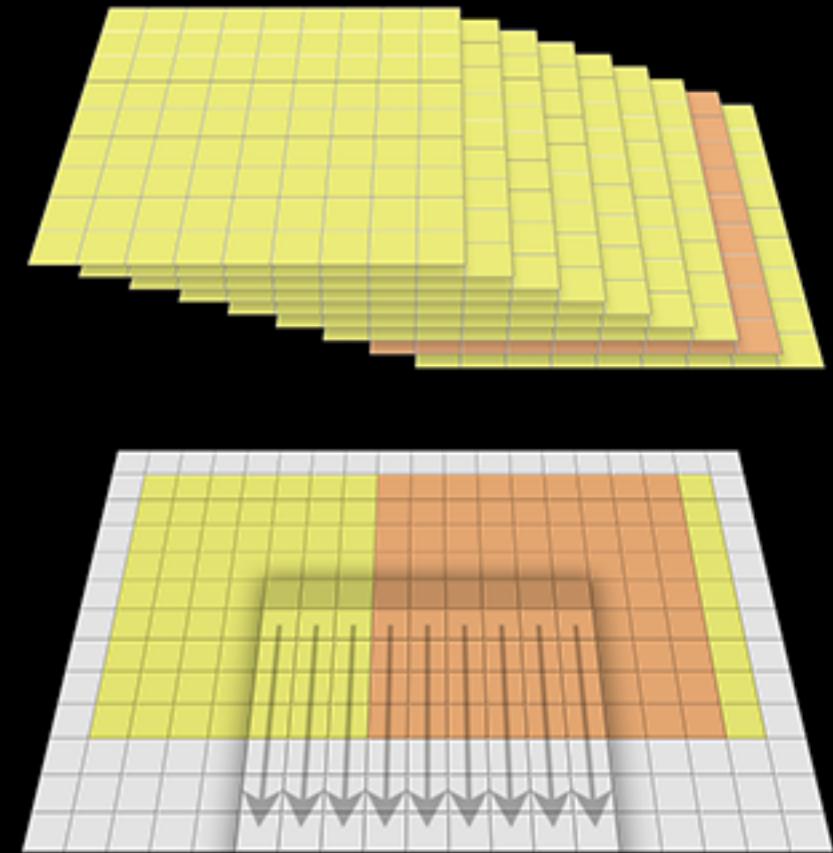
1	1	1	1	1	1	1
0	0	0	0	1	1	1
0	0	0	0	0	0	1
0	0	0	1	1	1	1
0	0	0	0	0	0	0
1	1	1	1	1	0	0
1	1	1	1	1	1	1
0	0	0	0	0	0	0
1	1	1	0	0	0	0
0	0	0	0	0	0	1



SIGGRAPH2006

# The Compound Histogram

- Redundancy in Binary  $H_n$



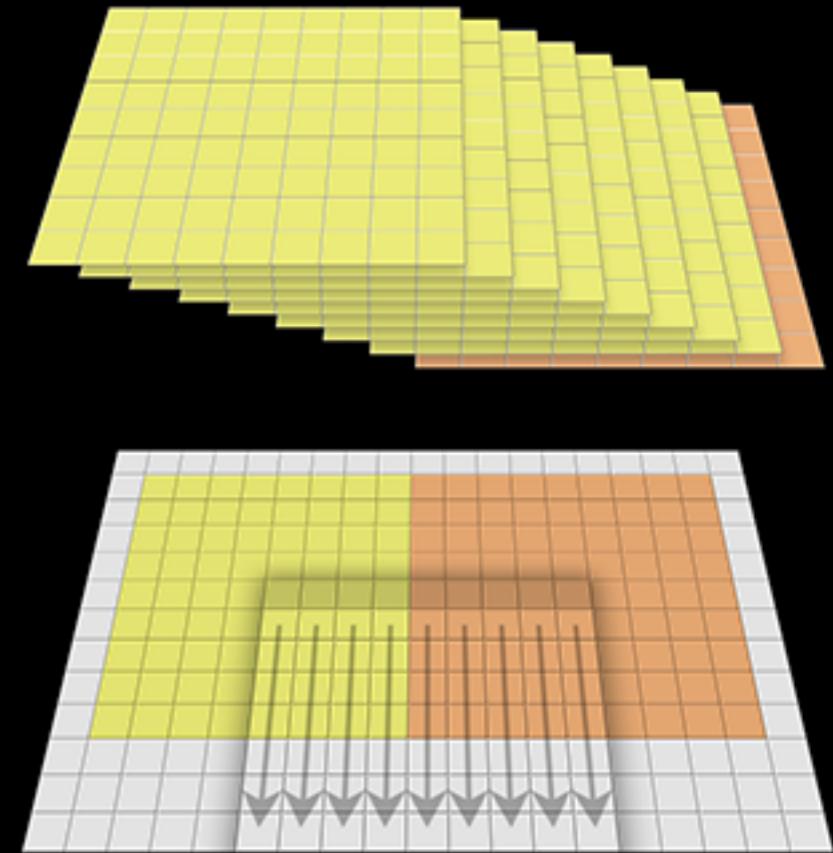
1	1	1	1	1	1	1
0	0	0	1	1	1	1
0	0	0	0	0	1	1
0	0	1	1	1	1	1
0	0	0	0	0	0	0
1	1	1	1	1	0	0
1	1	1	1	1	1	0
0	0	0	0	0	0	0
1	1	1	0	0	0	0
0	0	0	0	0	1	1



SIGGRAPH2006

# The Compound Histogram

- Redundancy in Binary  $H_n$



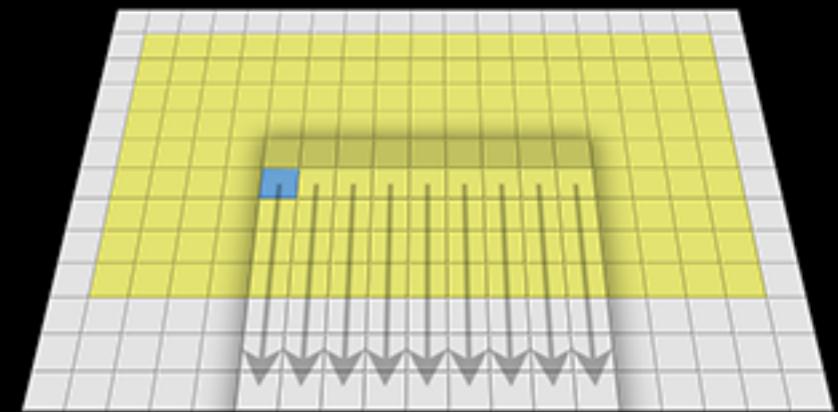
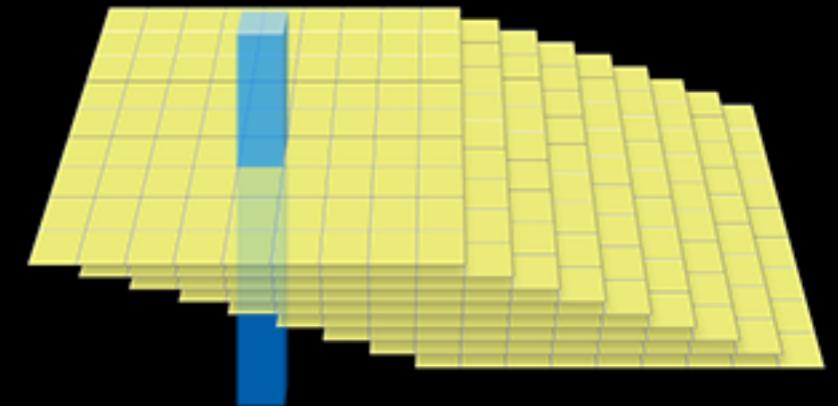
1	1	1	1	1	1	1	0
0	0	0	1	1	1	1	1
0	0	0	0	0	1	1	1
0	0	1	1	1	1	1	1
0	0	0	0	0	0	0	0
1	1	1	1	1	0	0	0
1	1	1	1	1	1	1	0
0	0	0	0	0	0	0	1
1	1	1	0	0	0	0	0
0	0	0	0	0	1	1	1



SIGGRAPH2006

# The Compound Histogram

- Redundancy in Binary  $H_n$



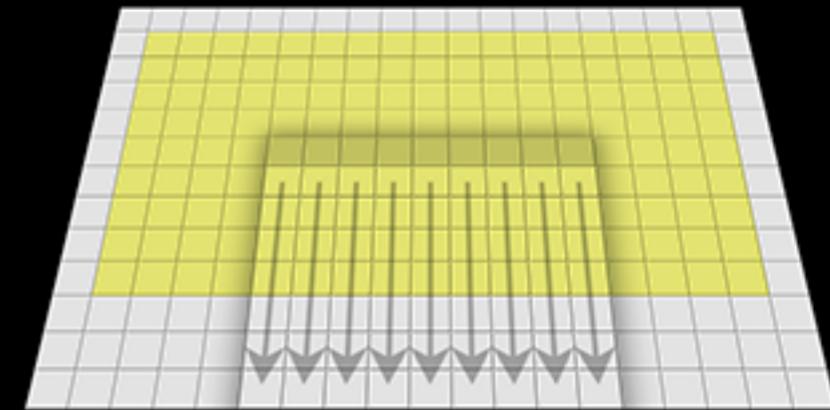
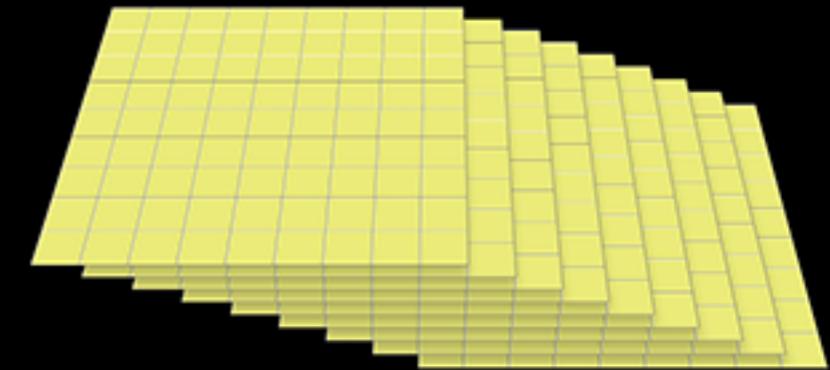
1	1	1	1	1	1	1	1	0
0	0	0	0	1	1	1	1	1
0	0	0	0	0	1	1	1	1
0	0	0	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0
1 1 1 1 1 0 0 0 0								
1	1	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0	1
1	1	1	0	0	0	0	0	0
0	0	0	0	0	1	1	1	1



SIGGRAPH2006

# The Compound Histogram

- Redundancy in Binary  $H_n$

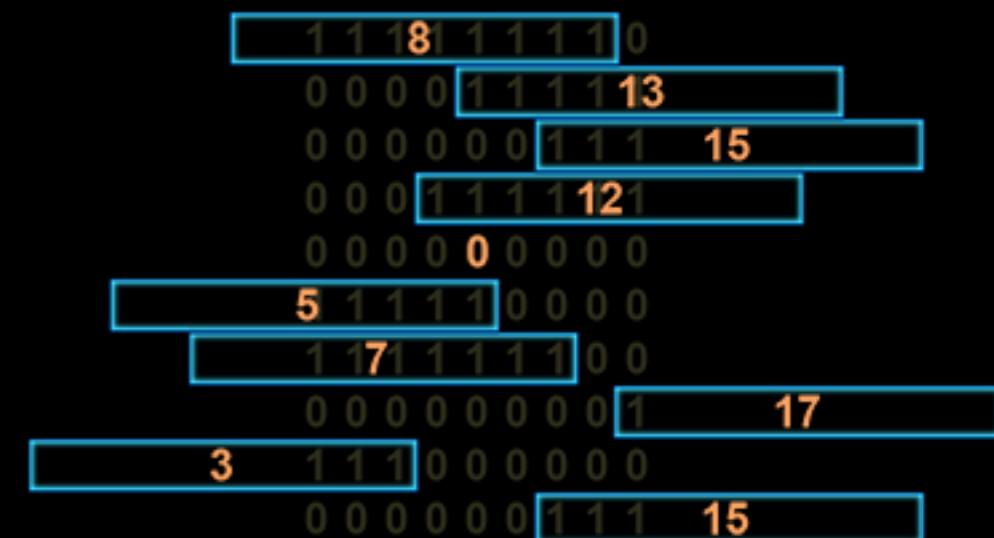
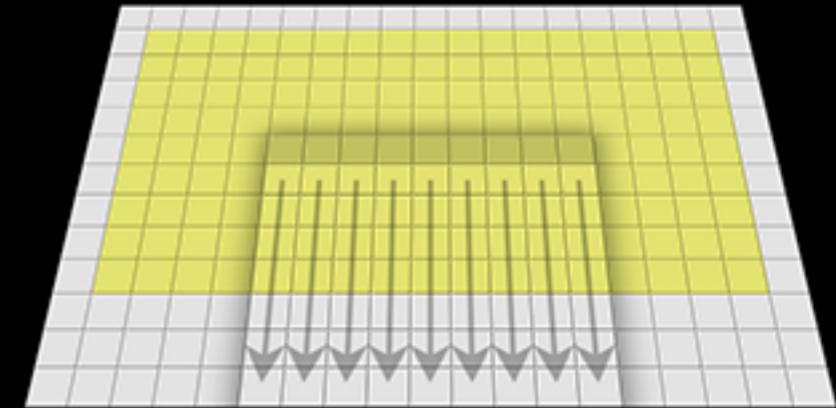
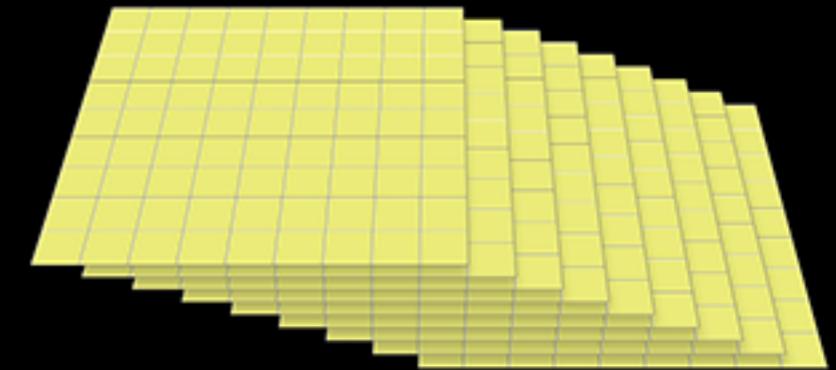




SIGGRAPH2006

# The Compound Histogram

- Redundancy in Binary  $H_n$

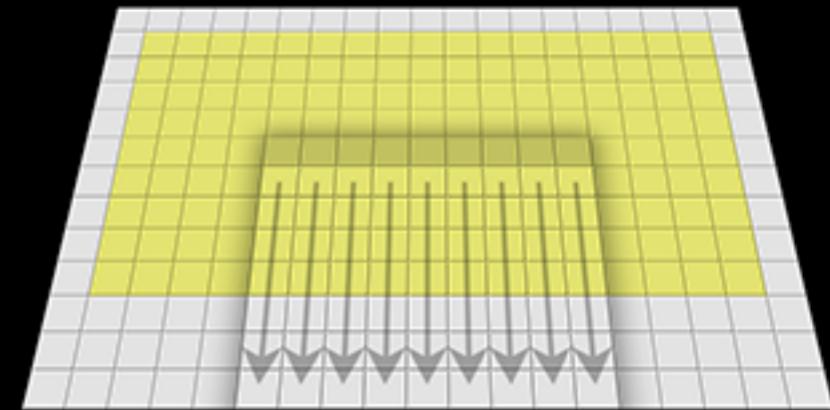
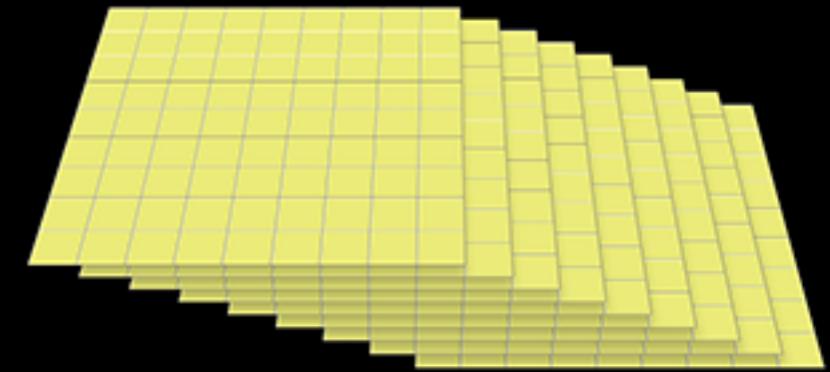




SIGGRAPH2006

# The Compound Histogram

- Redundancy in Binary  $H_n$
- Store Offsets in a “Compound” Histogram:  $H_c$



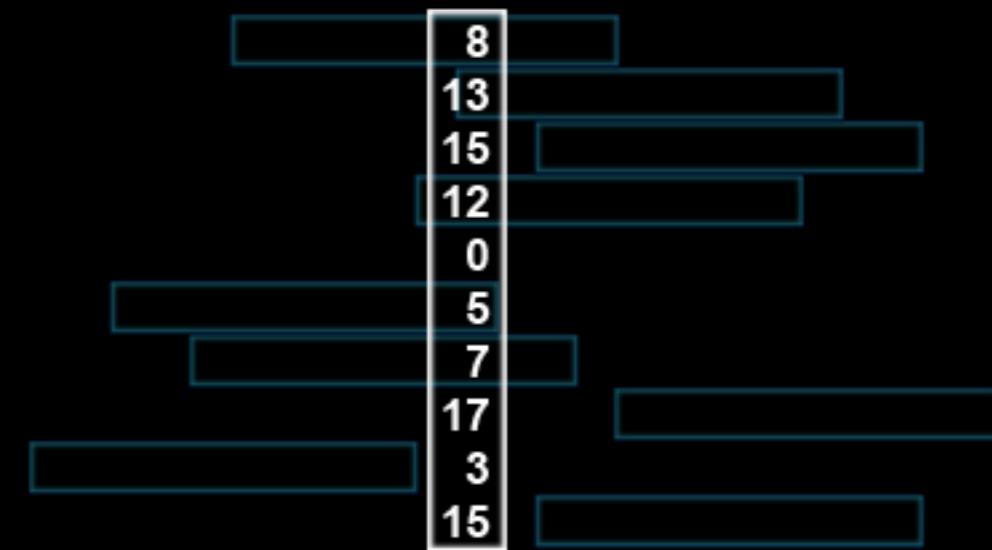
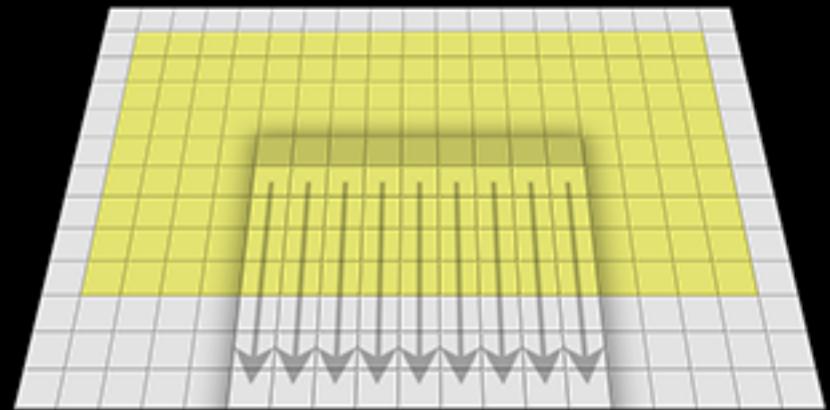
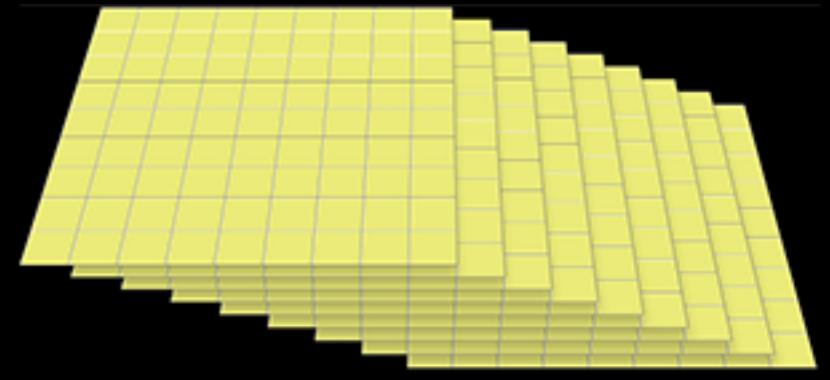
1	1	1	1	1	1	1	1	0
0	0	0	0	1	1	1	1	13
0	0	0	0	0	0	1	1	15
0	0	0	1	1	1	1	1	121
0	0	0	0	0	0	0	0	0
5	1	1	1	1	0	0	0	0
1	1	1	1	1	1	1	0	0
0	0	0	0	0	0	0	1	17
3	1	1	1	0	0	0	0	0
0	0	0	0	0	0	1	1	15



SIGGRAPH2006

# The Compound Histogram

- Redundancy in Binary  $H_n$
- Store Offsets in a  
“Compound” Histogram:  $H_c$

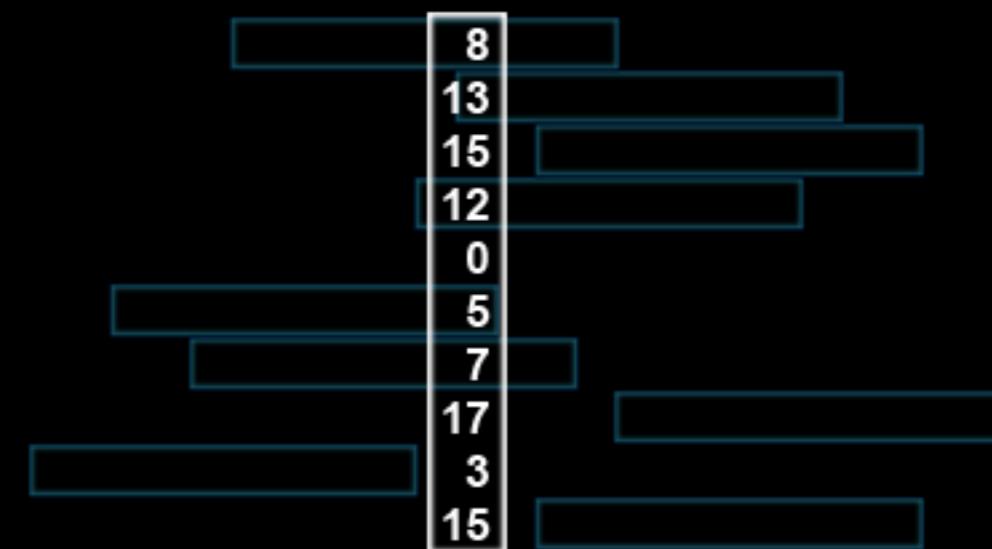
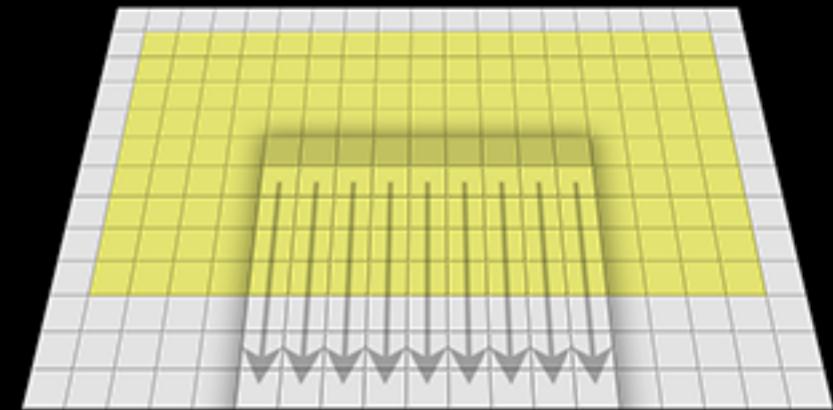
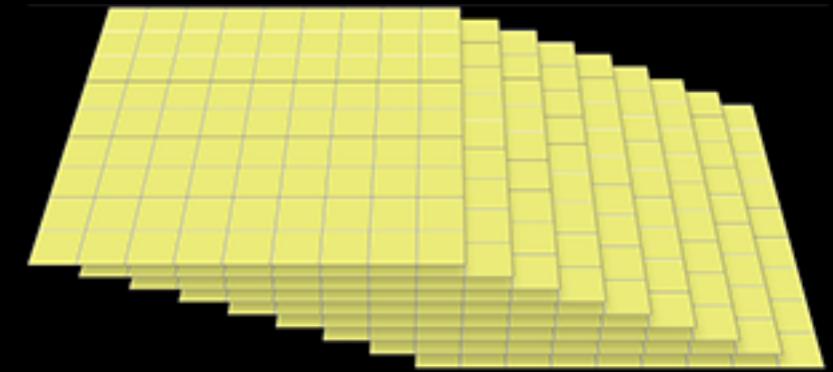


# The Compound Histogram



SIGGRAPH2006

- Redundancy in Binary  $H_n$
- Store Offsets in a “Compound” Histogram:  $H_c$
- $H_c \Rightarrow H_n$  in Constant Time!

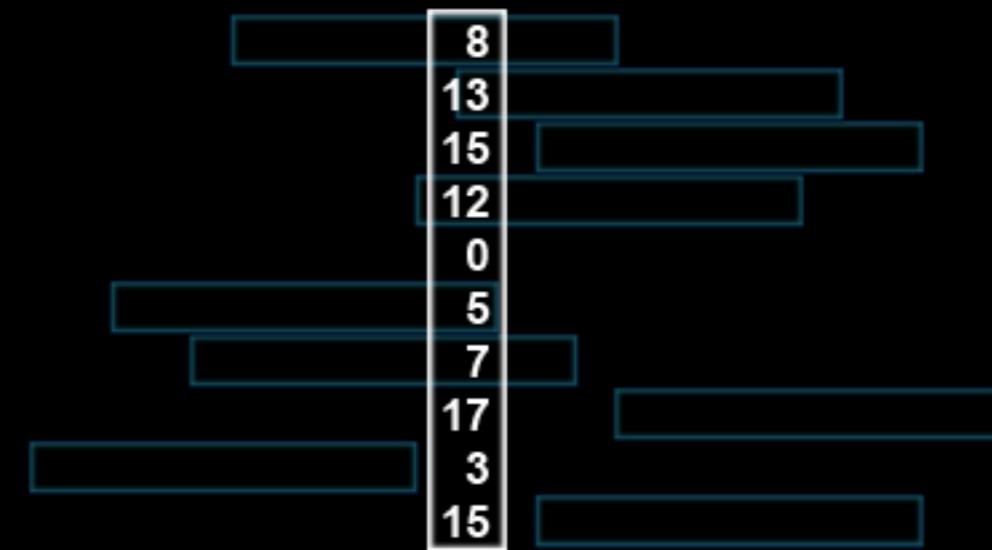
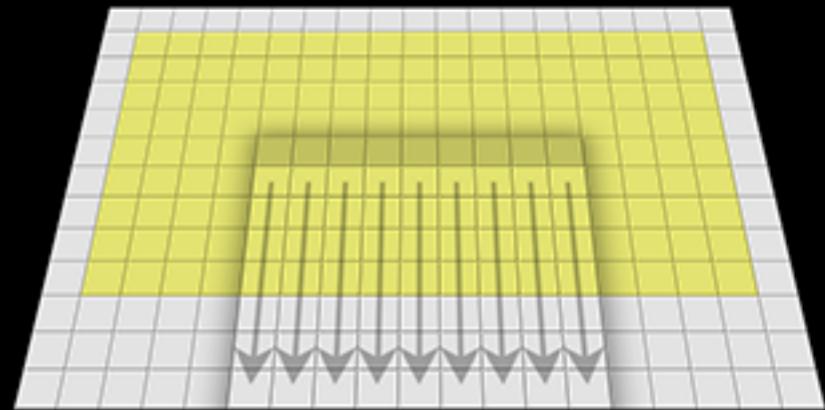
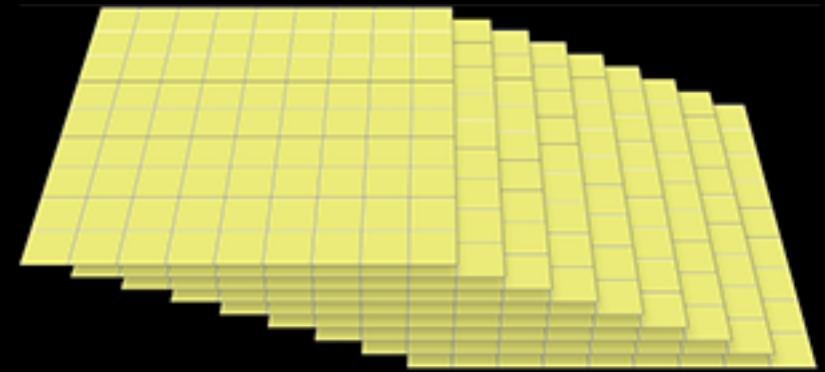


# The Compound Histogram



SIGGRAPH2006

- Redundancy in Binary  $H_n$
- Store Offsets in a “Compound” Histogram:  $H_c$
- $H_c \Rightarrow H_n$  in Constant Time!
- 8-Bit  $H_c \Leftrightarrow 128$  Columns

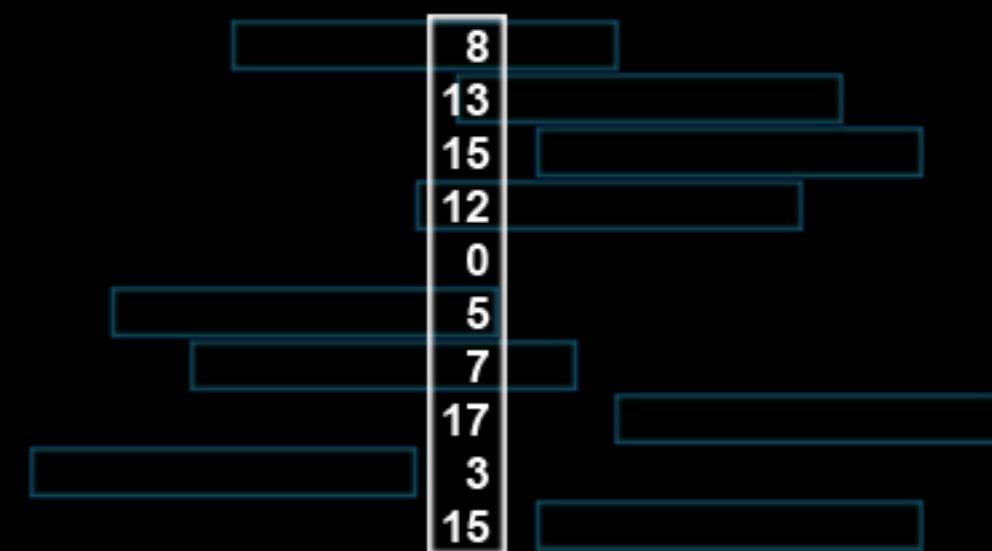
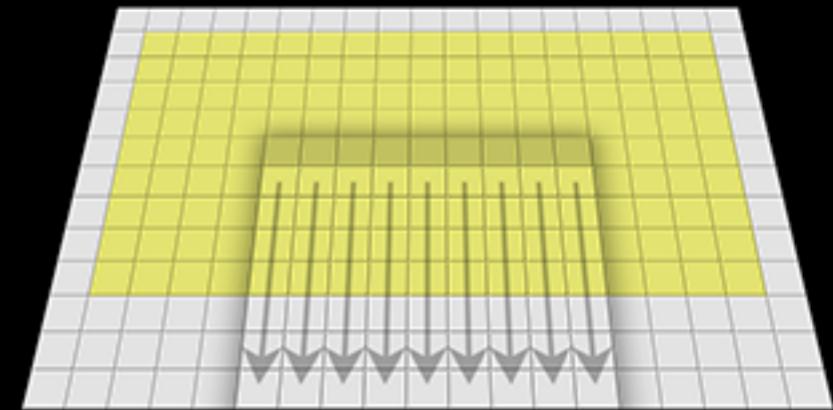
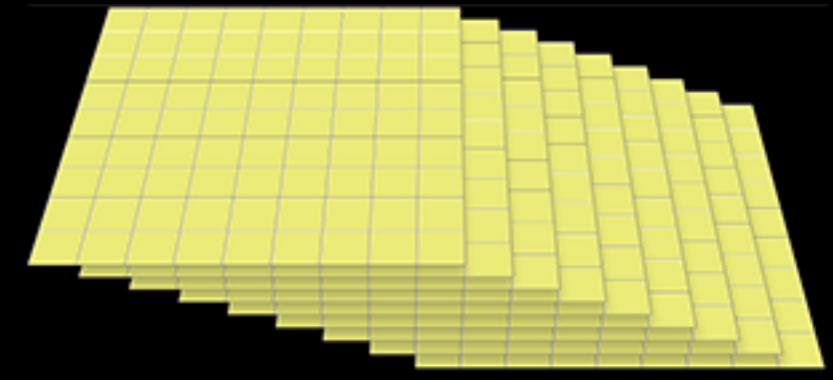


# The Compound Histogram



SIGGRAPH2006

- Redundancy in Binary  $H_n$
- Store Offsets in a “Compound” Histogram:  $H_c$
- $H_c \Rightarrow H_n$  in Constant Time!
- 8-Bit  $H_c \Leftrightarrow$  128 Columns
- Hybrid Algorithm:

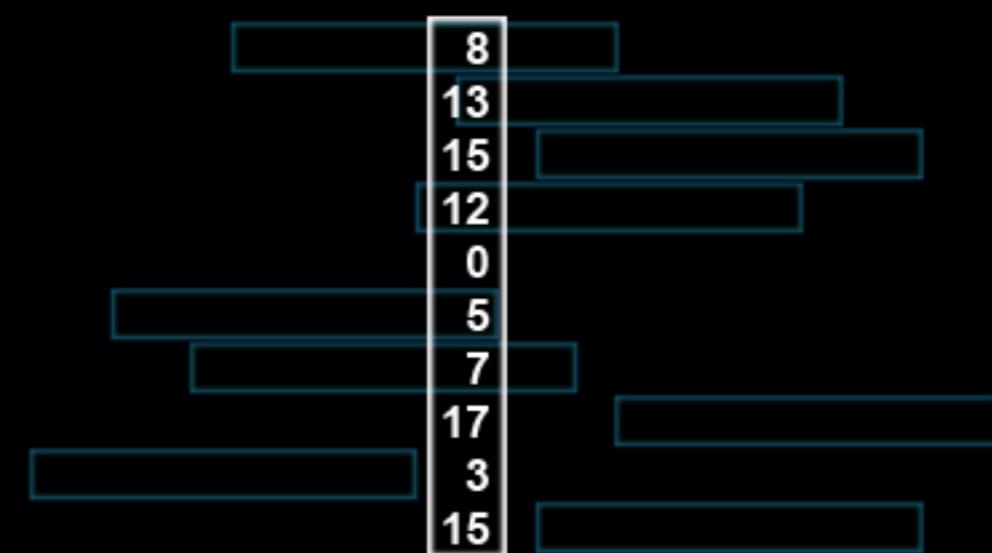
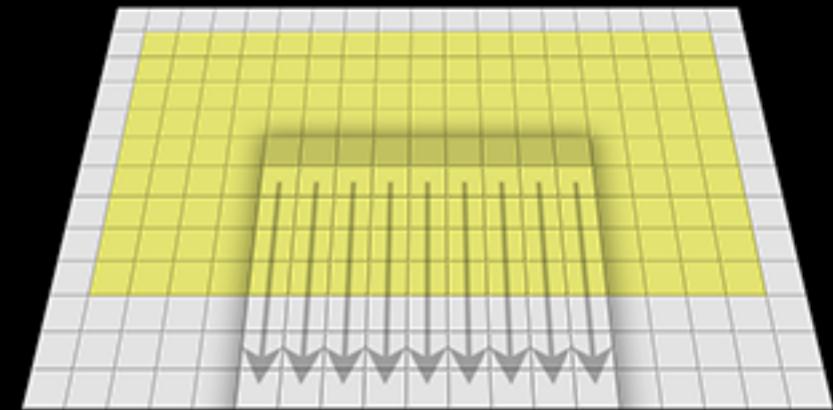
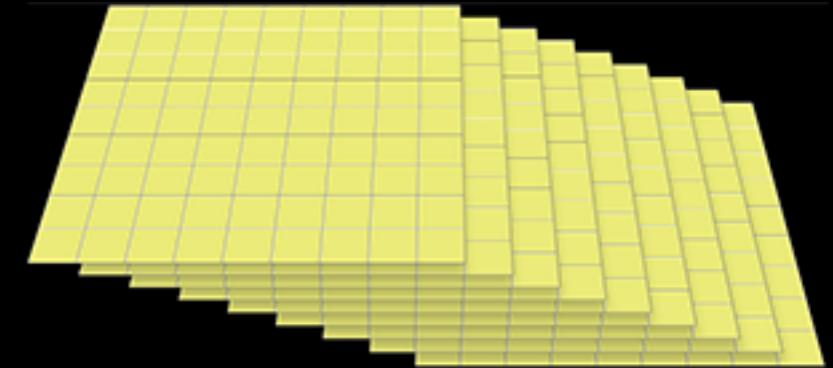


# The Compound Histogram



SIGGRAPH2006

- Redundancy in Binary  $H_n$
- Store Offsets in a “Compound” Histogram:  $H_c$
- $H_c \Rightarrow H_n$  in Constant Time!
- 8-Bit  $H_c \Leftrightarrow$  128 Columns
- Hybrid Algorithm:
  - 8-Bit Algorithm on High Bits

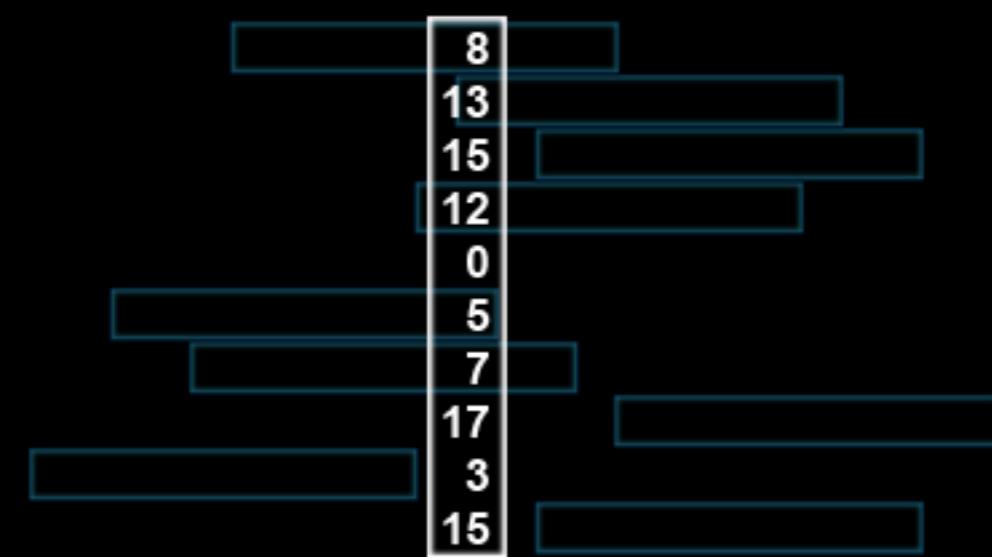
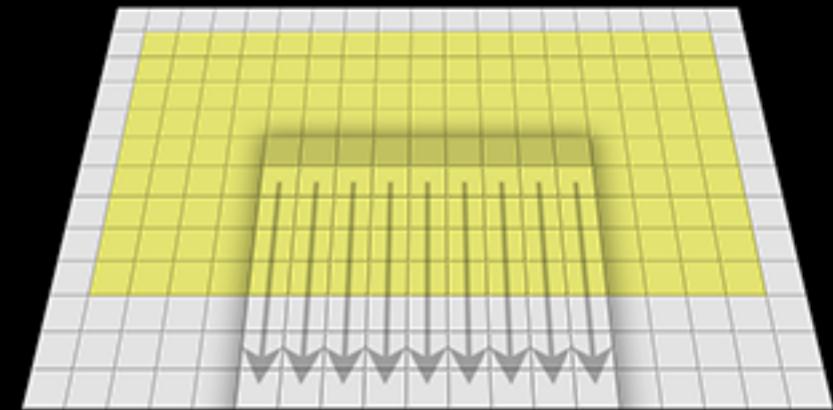
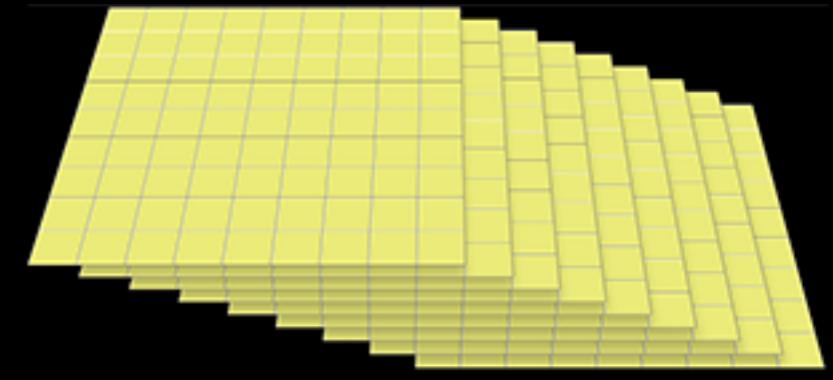


# The Compound Histogram



SIGGRAPH2006

- Redundancy in Binary  $H_n$
- Store Offsets in a “Compound” Histogram:  $H_c$
- $H_c \Rightarrow H_n$  in Constant Time!
- 8-Bit  $H_c \Leftrightarrow$  128 Columns
- Hybrid Algorithm:
  - 8-Bit Algorithm on High Bits
  - $H_c$  for Fine Tuning

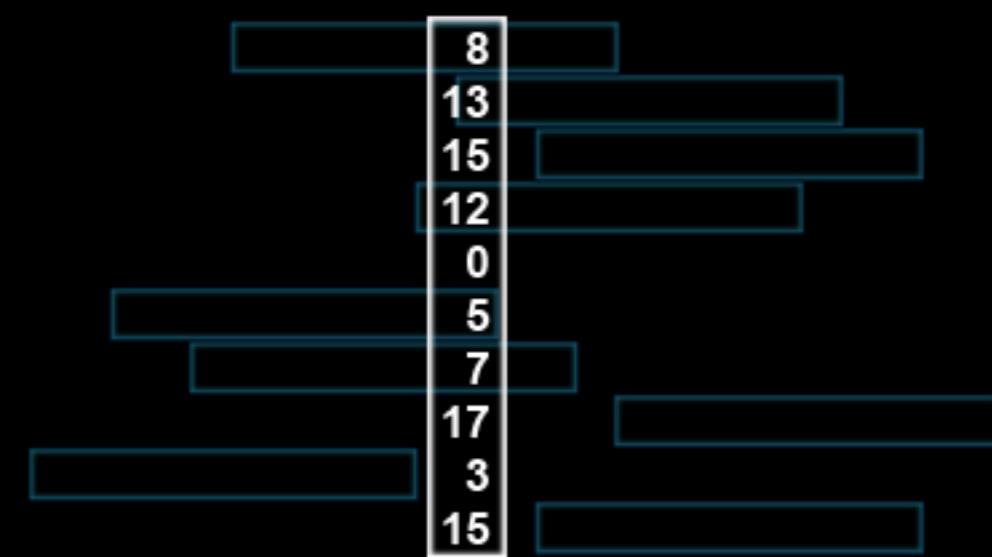
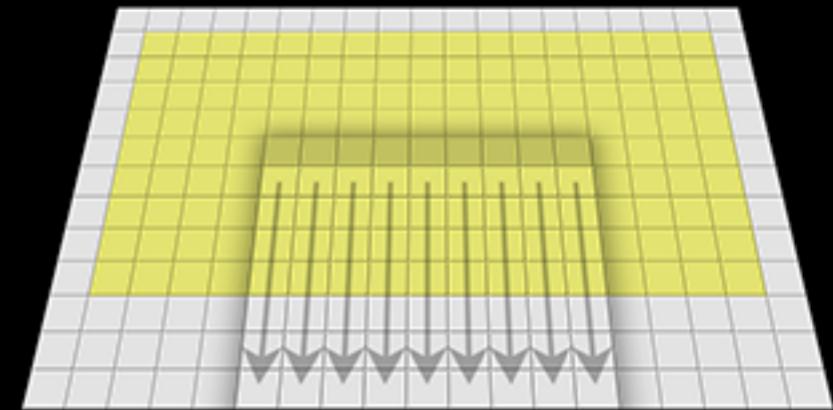
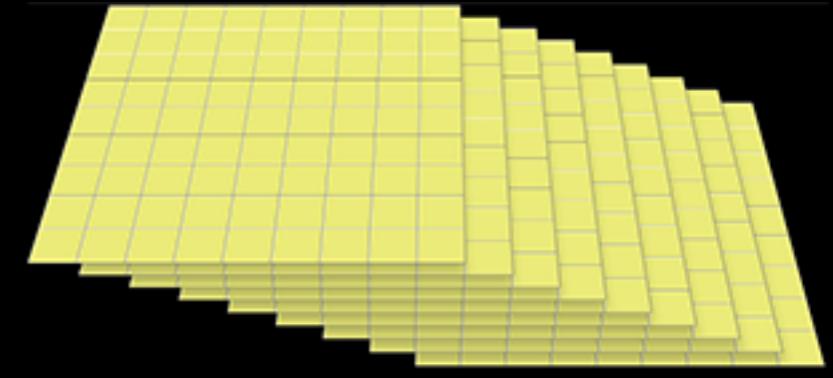


# The Compound Histogram

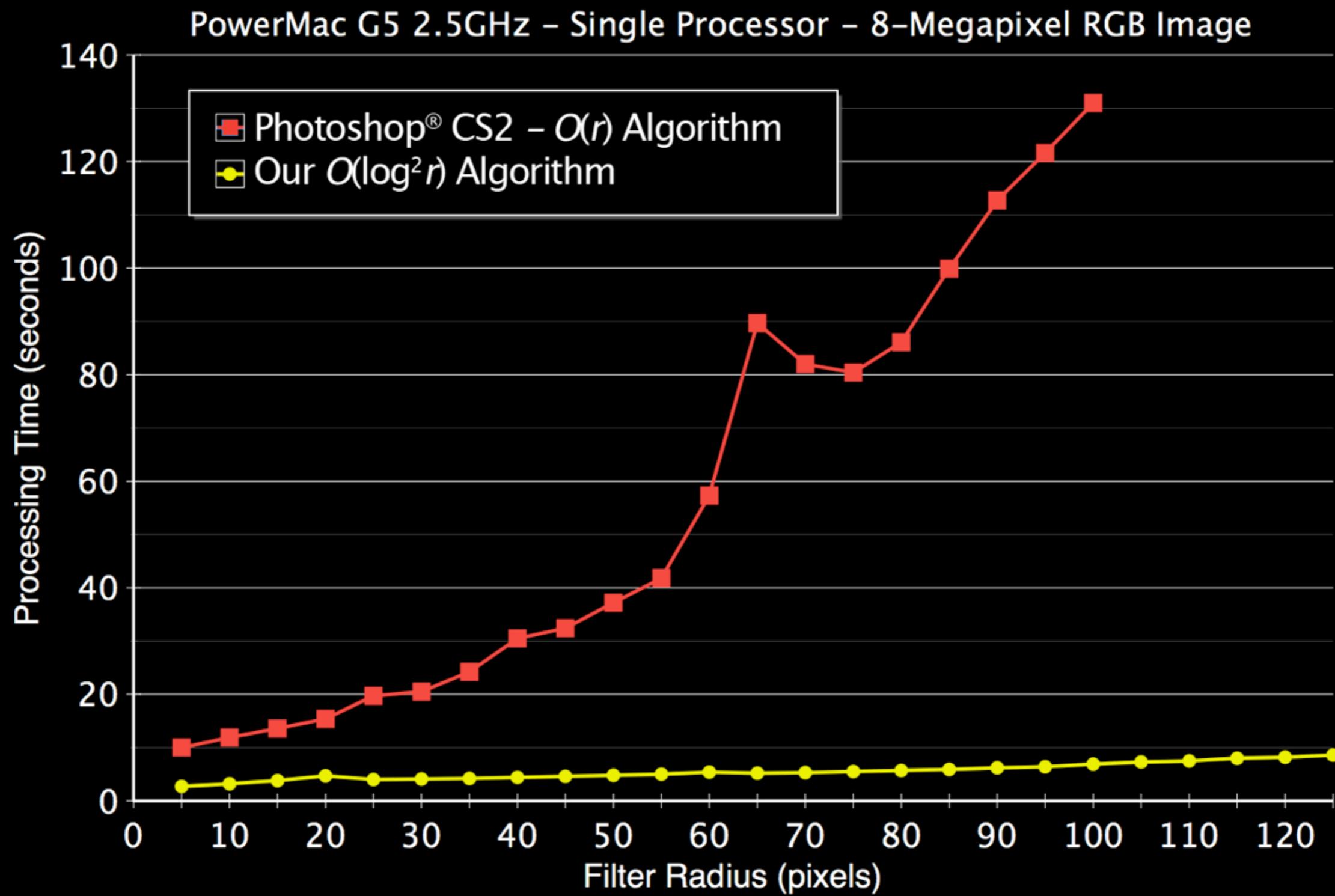


SIGGRAPH2006

- Redundancy in Binary  $H_n$
- Store Offsets in a “Compound” Histogram:  $H_c$
- $H_c \Rightarrow H_n$  in Constant Time!
- 8-Bit  $H_c \Leftrightarrow 128$  Columns
- Hybrid Algorithm:
  - 8-Bit Algorithm on High Bits
  - $H_c$  for Fine Tuning
  - Overall  $O(\log^2 r)$  Complexity



# 16-Bit Median - Performance



# Bilateral Filtering



SIGGRAPH2006

# Bilateral Filtering



SIGGRAPH2006

- Nonlinear Weighted Convolution



# Bilateral Filtering



SIGGRAPH2006

- Nonlinear Weighted Convolution
  - Bilateral = Blur that Favors “Similar” Values



# Bilateral Filtering



SIGGRAPH2006

- Nonlinear Weighted Convolution
  - Bilateral = Blur that Favors “Similar” Values
  - Weighted by Spatial and Intensity Difference



# Bilateral Filtering



SIGGRAPH2006

- Nonlinear Weighted Convolution
  - Bilateral = Blur that Favors “Similar” Values
  - Weighted by Spatial and Intensity Difference
- $$J_s = \sum_{p \in \Omega} f(p - s)g(I_p - I_s)I_p \Bigg/ \sum_{p \in \Omega} f(p - s)g(I_p - I_s)$$

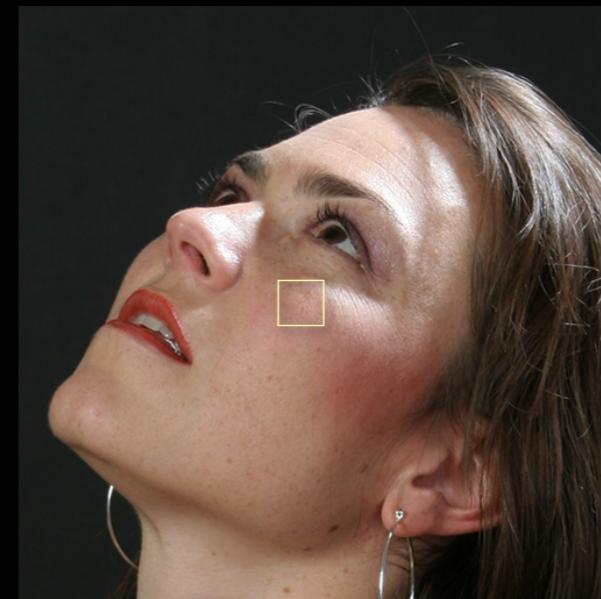


# Bilateral Filtering



SIGGRAPH2006

- Nonlinear Weighted Convolution
  - Bilateral = Blur that Favors “Similar” Values
  - Weighted by Spatial and Intensity Difference
- $$J_s = \sum_{p \in \Omega} f(p - s)g(I_p - I_s)I_p \Bigg/ \sum_{p \in \Omega} f(p - s)g(I_p - I_s)$$

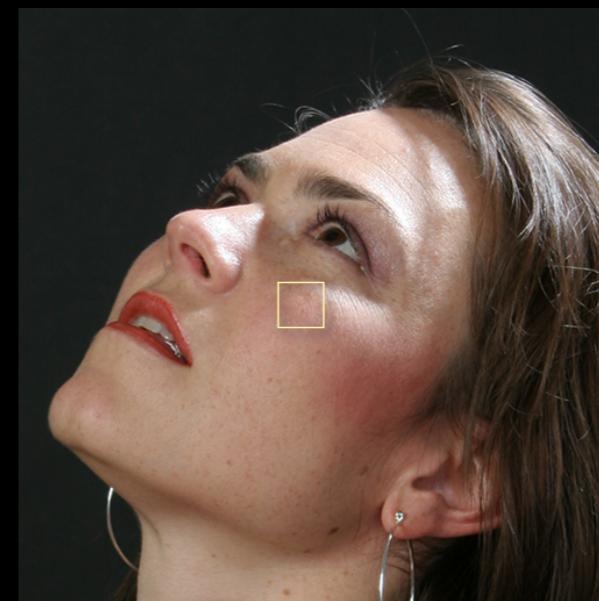


# Bilateral Filtering



SIGGRAPH2006

- Nonlinear Weighted Convolution
  - Bilateral = Blur that Favors “Similar” Values
  - Weighted by Spatial and Intensity Difference
- $$J_s = \sum_{p \in \Omega} f(p - s)g(I_p - I_s)I_p \Bigg/ \sum_{p \in \Omega} f(p - s)g(I_p - I_s)$$



# Bilateral Filtering



SIGGRAPH2006

- Nonlinear Weighted Convolution
  - Bilateral = Blur that Favors “Similar” Values
  - Weighted by Spatial and Intensity Difference
- $$J_s = \sum_{p \in \Omega} f(p - s)g(I_p - I_s)I_p \Bigg/ \sum_{p \in \Omega} f(p - s)g(I_p - I_s)$$





SIGGRAPH2006

# Bilateral Filtering

- Nonlinear Weighted Convolution
  - Bilateral = Blur that Favors “Similar” Values
  - Weighted by Spatial and Intensity Difference
- $$J_s = \sum_{p \in \Omega} f(p - s)g(I_p - I_s)I_p \Bigg/ \sum_{p \in \Omega} f(p - s)g(I_p - I_s)$$



# Bilateral Filtering



SIGGRAPH2006

- Nonlinear Weighted Convolution
  - Bilateral = Blur that Favors “Similar” Values
  - Weighted by Spatial and Intensity Difference
- $$J_s = \sum_{p \in \Omega} f(p - s)g(I_p - I_s)I_p \Bigg/ \sum_{p \in \Omega} f(p - s)g(I_p - I_s)$$
  - $f()$  and  $g()$  are typically Gaussian



# Bilateral Filtering



SIGGRAPH2006

- Nonlinear Weighted Convolution
  - Bilateral = Blur that Favors “Similar” Values
  - Weighted by Spatial and Intensity Difference
- $$J_s = \sum_{p \in \Omega} f(p - s)g(I_p - I_s)I_p \Bigg/ \sum_{p \in \Omega} f(p - s)g(I_p - I_s)$$
  - $f()$  and  $g()$  are typically Gaussian
- Special Case - Box Filter



# Bilateral Filtering



SIGGRAPH2006

- Nonlinear Weighted Convolution
  - Bilateral = Blur that Favors “Similar” Values
  - Weighted by Spatial and Intensity Difference
- $$J_s = \sum_{p \in \Omega} f(p - s)g(I_p - I_s)I_p \Bigg/ \sum_{p \in \Omega} f(p - s)g(I_p - I_s)$$
  - $f()$  and  $g()$  are typically Gaussian
- Special Case - Box Filter
- Relative Intensity



# Bilateral Filtering



SIGGRAPH2006

- Nonlinear Weighted Convolution
  - Bilateral = Blur that Favors “Similar” Values
  - Weighted by Spatial and Intensity Difference
- $$J_s = \sum_{p \in \Omega} f(p - s)g(I_p - I_s)I_p \Bigg/ \sum_{p \in \Omega} f(p - s)g(I_p - I_s)$$
  - $f()$  and  $g()$  are typically Gaussian
- Special Case - Box Filter
- Relative Intensity
- Smooth Spatial Falloff



# Special Case - Box Filter Bilateral



SIGGRAPH2006





SIGGRAPH2006

# Special Case - Box Filter Bilateral

$$\bullet \quad J_s = \sum_{p \in \Omega} f(p-s)g(I_p - I_s)I_p \Bigg/ \sum_{p \in \Omega} f(p-s)g(I_p - I_s)$$





SIGGRAPH2006

# Special Case - Box Filter Bilateral

- $$J_s = \sum_{p \in \Omega} f(p - s)g(I_p - I_s)I_p \Big/ \sum_{p \in \Omega} f(p - s)g(I_p - I_s)$$
- Box Filter - Uniform Spatial Weighting





SIGGRAPH2006

# Special Case - Box Filter Bilateral

- $J_s = \sum_{p \in \Omega} f(p-s)g(I_p - I_s)I_p \Big/ \sum_{p \in \Omega} f(p-s)g(I_p - I_s)$
- Box Filter - Uniform Spatial Weighting

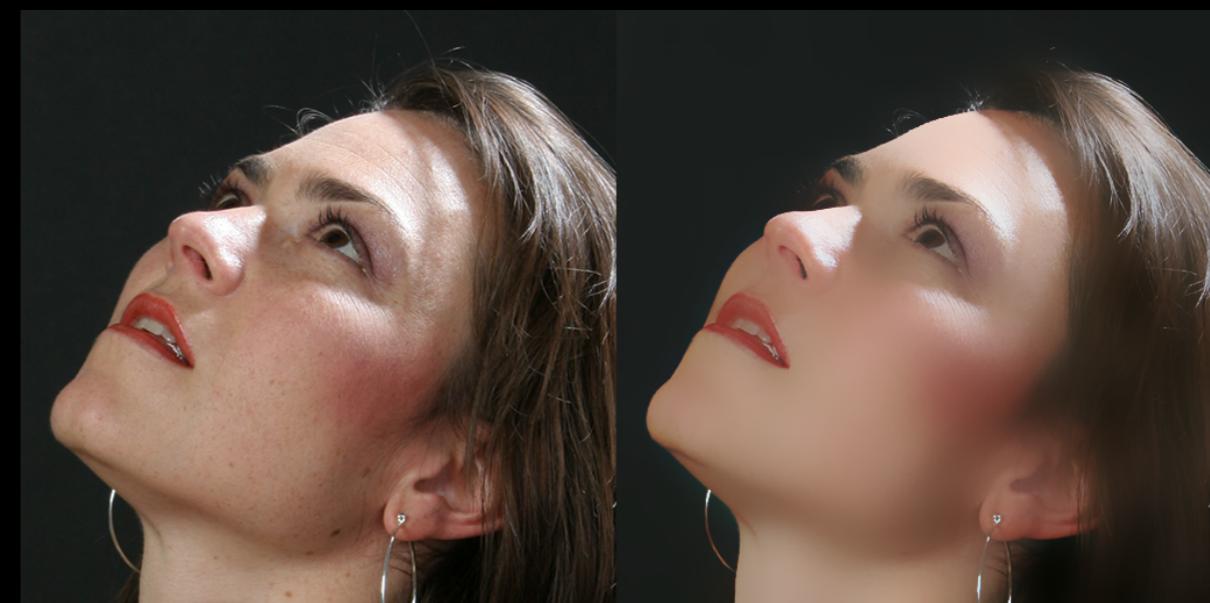




SIGGRAPH2006

# Special Case - Box Filter Bilateral

- $J_s = \sum_{p \in \Omega} f(p-s)g(I_p - I_s)I_p \Big/ \sum_{p \in \Omega} f(p-s)g(I_p - I_s)$
- Box Filter - Uniform Spatial Weighting
- $J_s = \sum_{p \in W} g(I_p - I_s)I_p \Big/ \sum_{p \in W} g(I_p - I_s)$





SIGGRAPH2006

# Special Case - Box Filter Bilateral

- $J_s = \sum_{p \in \Omega} f(p-s)g(I_p - I_s)I_p \Big/ \sum_{p \in \Omega} f(p-s)g(I_p - I_s)$
- Box Filter - Uniform Spatial Weighting
- $J_s = \sum_{p \in W} g(I_p - I_s)I_p \Big/ \sum_{p \in W} g(I_p - I_s)$
- Can be Computed from *Histograms*



SIGGRAPH2006

# Special Case - Box Filter Bilateral

- $J_s = \sum_{p \in \Omega} f(p-s)g(I_p - I_s)I_p \Big/ \sum_{p \in \Omega} f(p-s)g(I_p - I_s)$
- Box Filter - Uniform Spatial Weighting
- $J_s = \sum_{p \in W} g(I_p - I_s)I_p \Big/ \sum_{p \in W} g(I_p - I_s)$
- Can be Computed from *Histograms*
- $J_s = \sum_{v=0}^{255} H[v]g(v - I_s)v \Big/ \sum_{v=0}^{255} H[v]g(v - I_s)$



SIGGRAPH2006

# Special Case - Box Filter Bilateral

- $$J_s = \sum_{p \in \Omega} f(p-s)g(I_p - I_s)I_p \Bigg/ \sum_{p \in \Omega} f(p-s)g(I_p - I_s)$$
- Box Filter - Uniform Spatial Weighting
- $$J_s = \sum_{p \in W} g(I_p - I_s)I_p \Bigg/ \sum_{p \in W} g(I_p - I_s)$$
- Can be Computed from *Histograms*
- $$J_s = \sum_{v=0}^{255} H[v]g(v - I_s)v \Bigg/ \sum_{v=0}^{255} H[v]g(v - I_s)$$
- Use 8-Bit Median to Generate Histograms



SIGGRAPH2006

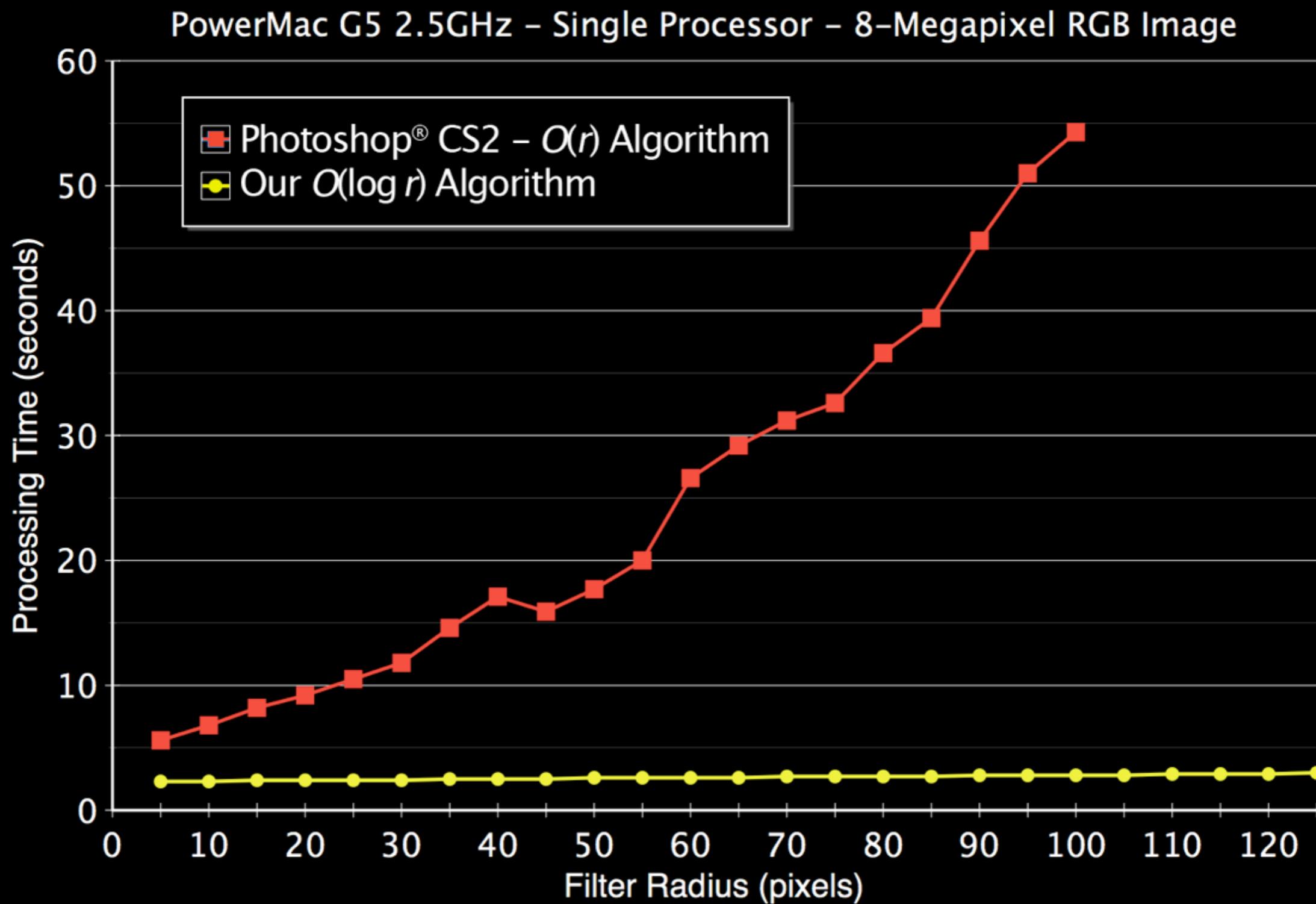
# Special Case - Box Filter Bilateral

- $J_s = \sum_{p \in \Omega} f(p-s)g(I_p - I_s)I_p \Big/ \sum_{p \in \Omega} f(p-s)g(I_p - I_s)$
- Box Filter - Uniform Spatial Weighting
- $J_s = \sum_{p \in W} g(I_p - I_s)I_p \Big/ \sum_{p \in W} g(I_p - I_s)$
- Can be Computed from *Histograms*
- $J_s = \sum_{v=0}^{255} H[v]g(v - I_s)v \Big/ \sum_{v=0}^{255} H[v]g(v - I_s)$
- Use 8-Bit Median to Generate Histograms
- For higher-precision data, blend into 8 bit



SIGGRAPH2006

# Bilateral Filter - Performance



# Relative Intensity Bilateral



SIGGRAPH2006

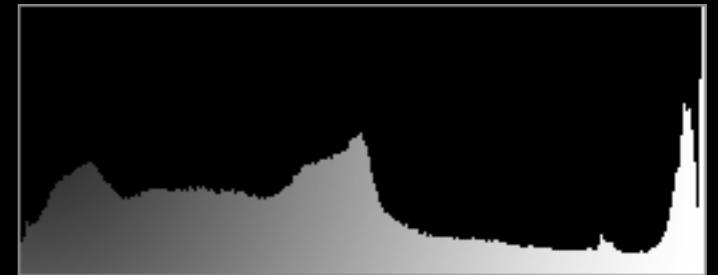


# Relative Intensity Bilateral



SIGGRAPH2006

- The Eye Perceives Relative Brightness

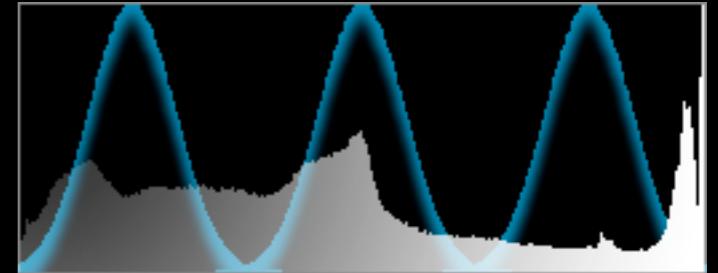


# Relative Intensity Bilateral



SIGGRAPH2006

- The Eye Perceives Relative Brightness

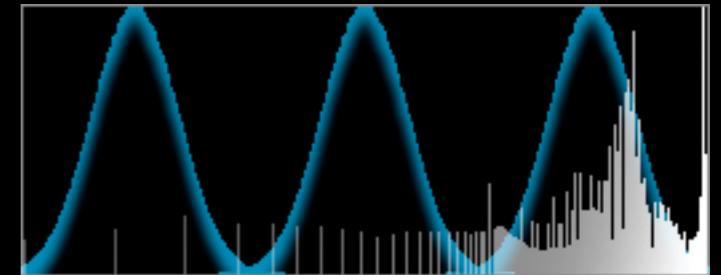


# Relative Intensity Bilateral



SIGGRAPH2006

- The Eye Perceives Relative Brightness
- Logarithmic Mapping

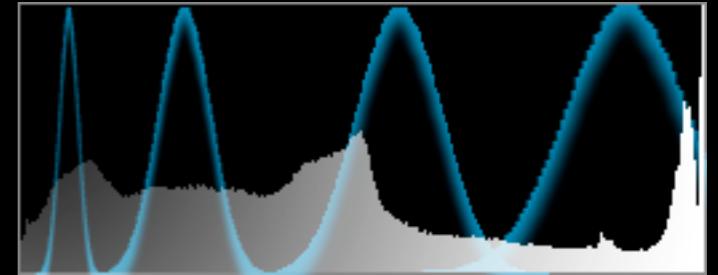


# Relative Intensity Bilateral



SIGGRAPH2006

- The Eye Perceives Relative Brightness
- Logarithmic Mapping
- Scaled Intensity Function

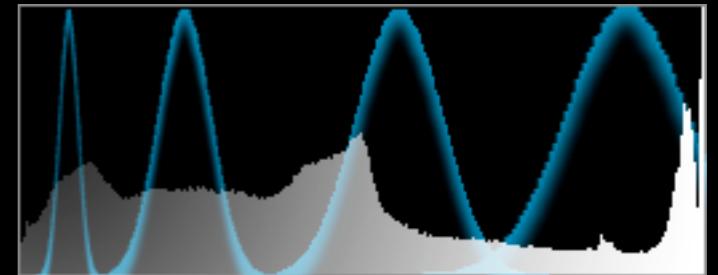


# Relative Intensity Bilateral



SIGGRAPH2006

- The Eye Perceives Relative Brightness
- Logarithmic Mapping
- Scaled Intensity Function
  - Width Proportional to Brightness

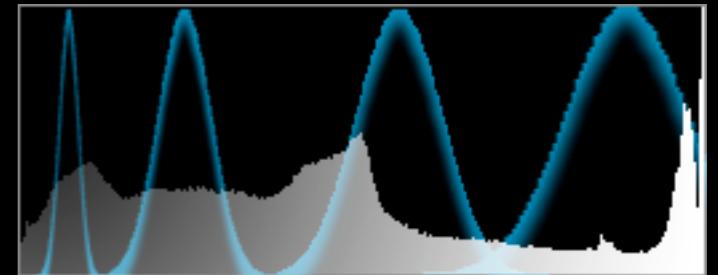


# Relative Intensity Bilateral



SIGGRAPH2006

- The Eye Perceives Relative Brightness
- Logarithmic Mapping
- Scaled Intensity Function
  - Width Proportional to Brightness



# Bilateral Filter



SIGGRAPH2006

---

Demo

# String Theory



SIGGRAPH2006

---

# String Theory



SIGGRAPH2006

BOS 6  
2 TON

# String Theory



SIGGRAPH2006

BOSS  
2TON

# String Theory



SIGGRAPH2006



# String Theory



SIGGRAPH2006



# String Theory



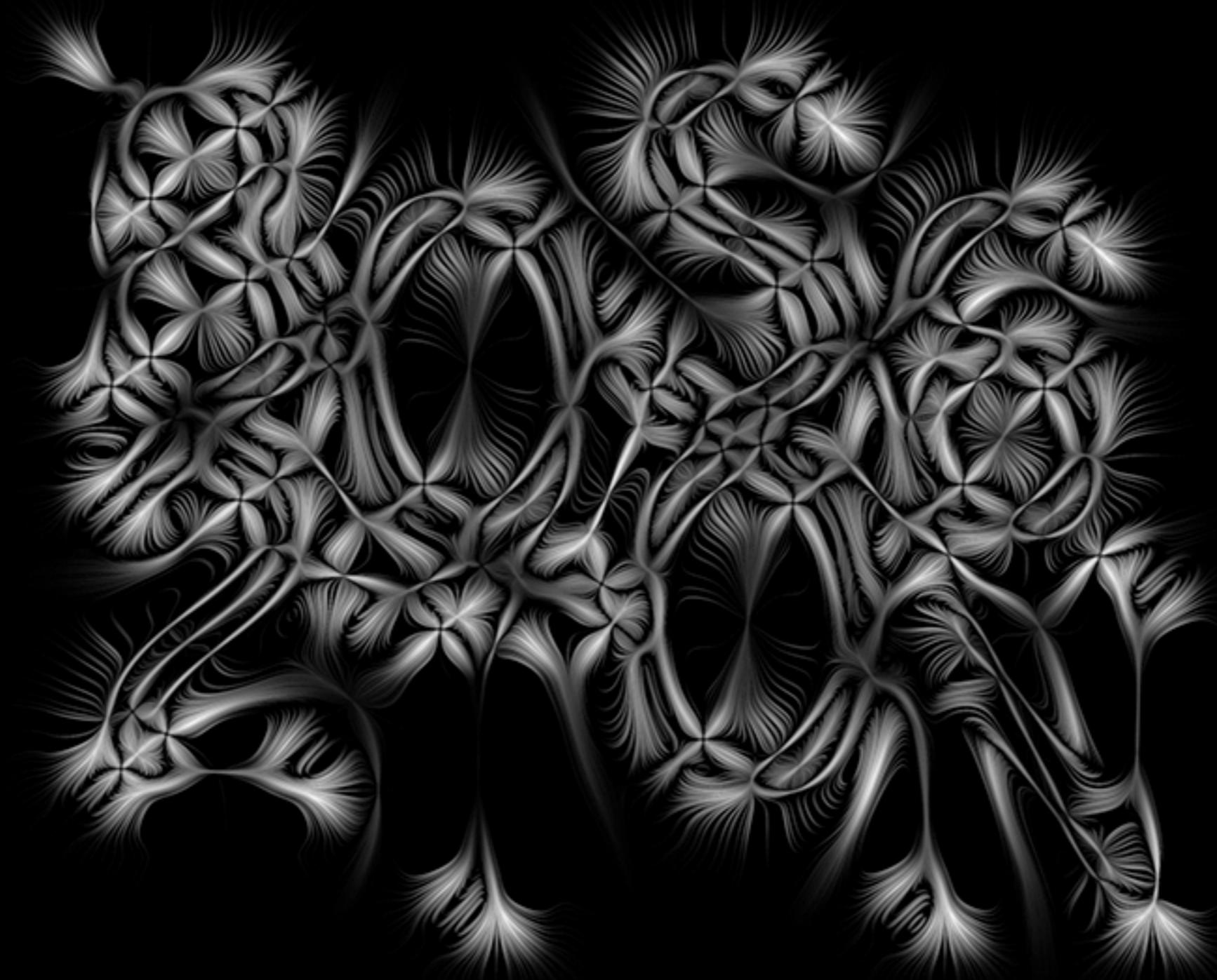
SIGGRAPH2006



# String Theory



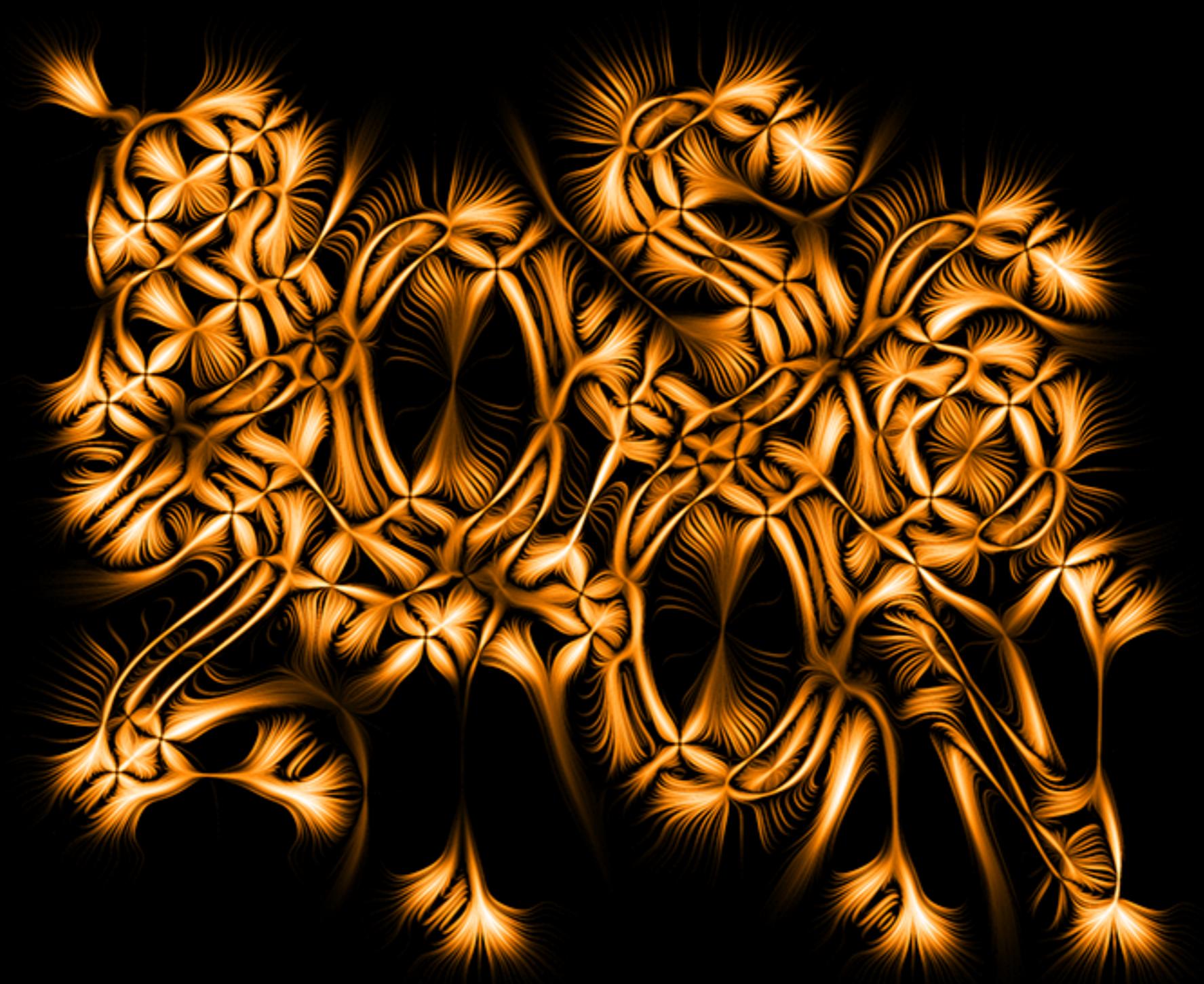
SIGGRAPH2006



# String Theory



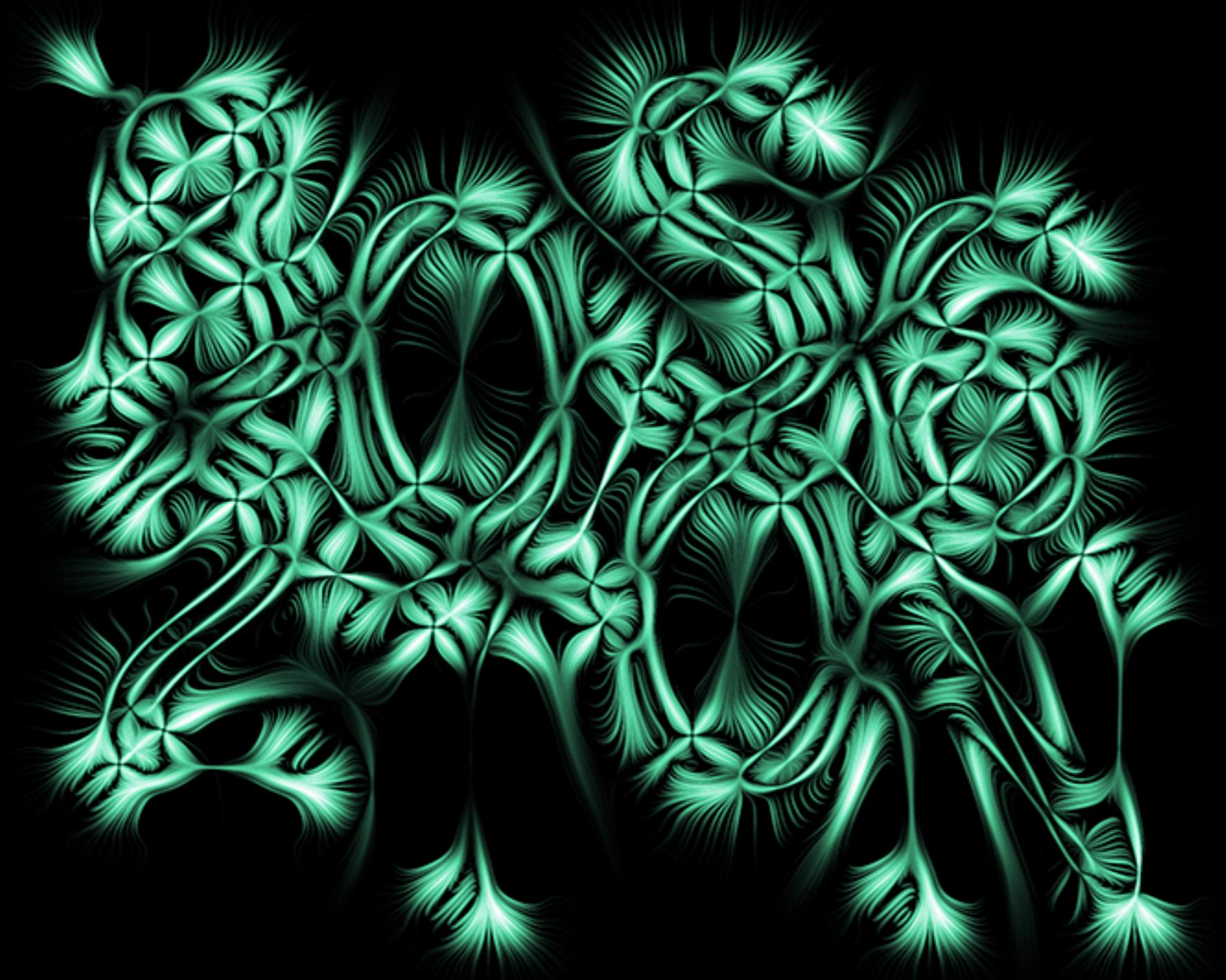
SIGGRAPH2006



# String Theory



SIGGRAPH2006



# Conclusion



SIGGRAPH2006



SIGGRAPH2006

# Conclusion

- Fast 2D Median Filtering Algorithms



SIGGRAPH2006

# Conclusion

---

- Fast 2D Median Filtering Algorithms
- Efficient Bilateral Filtering Algorithm

# Conclusion



SIGGRAPH2006

- 
- Fast 2D Median Filtering Algorithms
  - Efficient Bilateral Filtering Algorithm
  - Future Directions

# Conclusion



SIGGRAPH2006

- Fast 2D Median Filtering Algorithms
- Efficient Bilateral Filtering Algorithm
- Future Directions
  - Theoretical Analysis

# Conclusion



SIGGRAPH2006

- Fast 2D Median Filtering Algorithms
- Efficient Bilateral Filtering Algorithm
- Future Directions
  - Theoretical Analysis
  - GPU Implementation

# Conclusion



SIGGRAPH2006

- Fast 2D Median Filtering Algorithms
- Efficient Bilateral Filtering Algorithm
- Future Directions
  - Theoretical Analysis
  - GPU Implementation
  - 3D Median, Variable Width, Circular, etc.



SIGGRAPH2006

# Conclusion

---

- Fast 2D Median Filtering Algorithms
- Efficient Bilateral Filtering Algorithm
- Future Directions
  - Theoretical Analysis
  - GPU Implementation
  - 3D Median, Variable Width, Circular, etc.

contact: [ben@shellandslate.com](mailto:ben@shellandslate.com)

<http://www.shellandslate.com/fastmedian.html>