3D Gaussian Splatting

Tino Reith

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Dret- takis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Render- ing. ACM Trans. Graph. 42, 4, Article 1 (August 2023), 14 pages. https: //doi.org/10.1145/3592433

(Demo 1)

3D Gaussian Splatting

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Dret- takis. 2023. 3D Gaussian Splatting for Real-Time Radiance Field Render- ing. ACM Trans. Graph. 42, 4, Article 1 (August 2023), 14 pages. https: //doi.org/10.1145/3592433

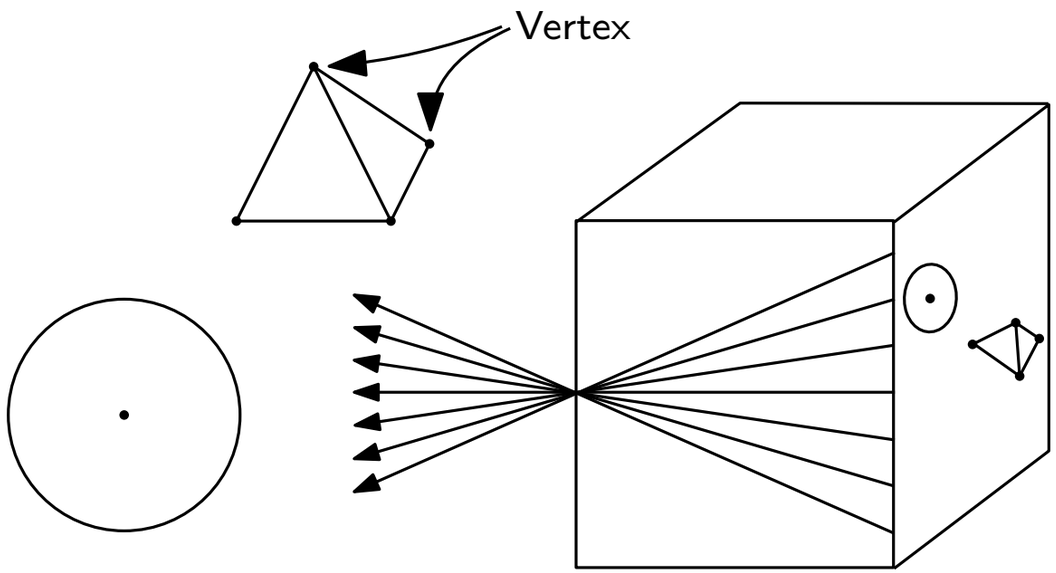
Traditionelle Rendering-Techniken

1) Raytracing

2) Rasterization

Traditionelle Rendering-Techniken

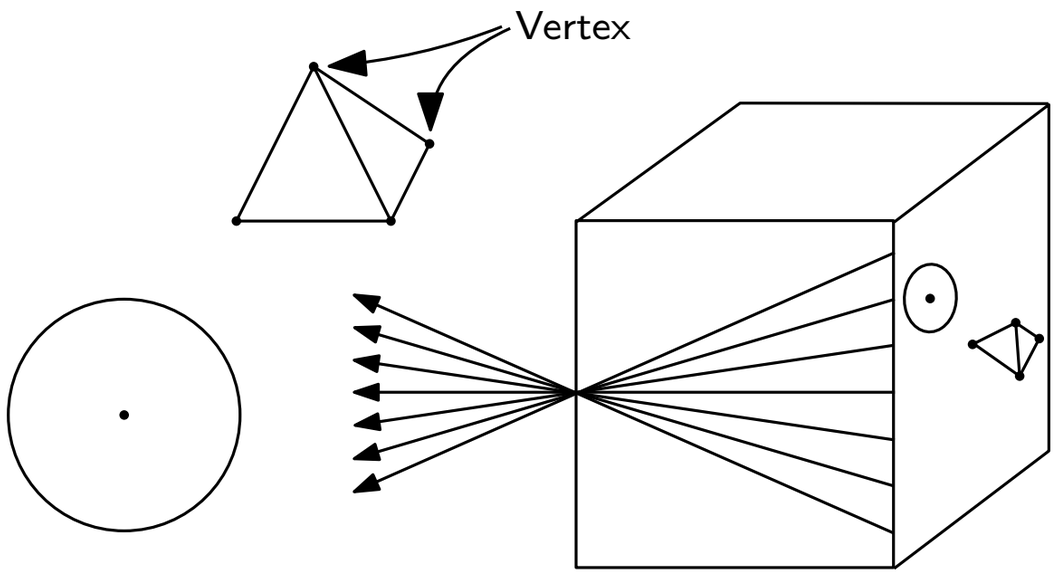
1) Raytracing



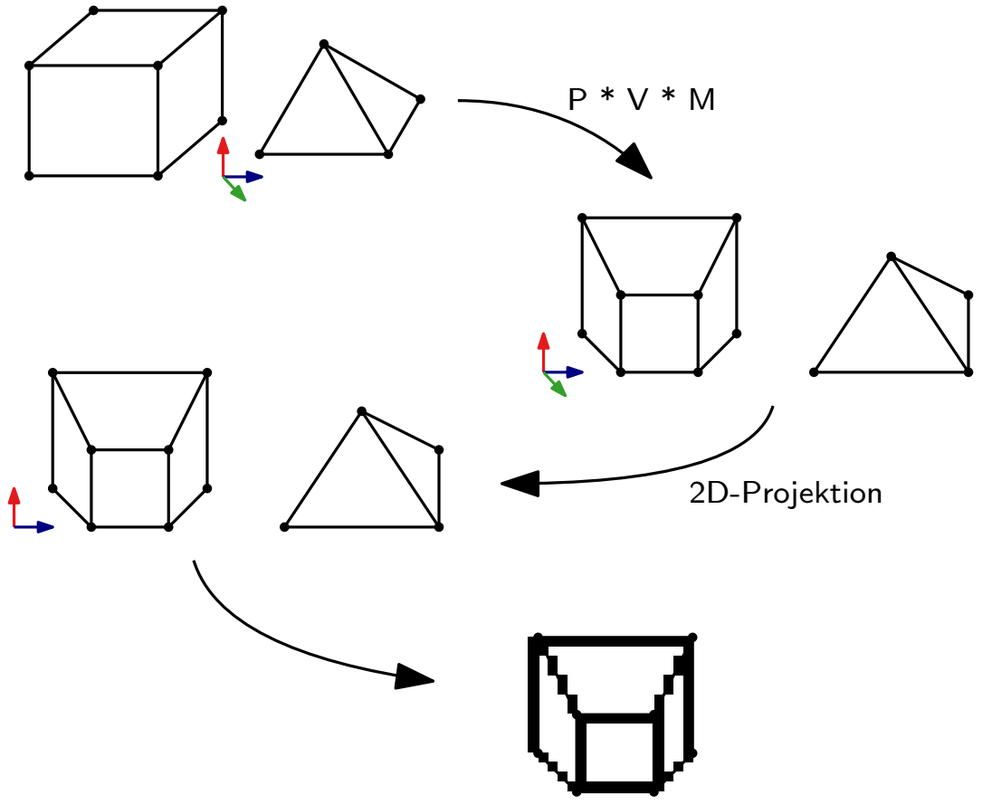
2) Rasterization

Traditionelle Rendering-Techniken

1) Raytracing

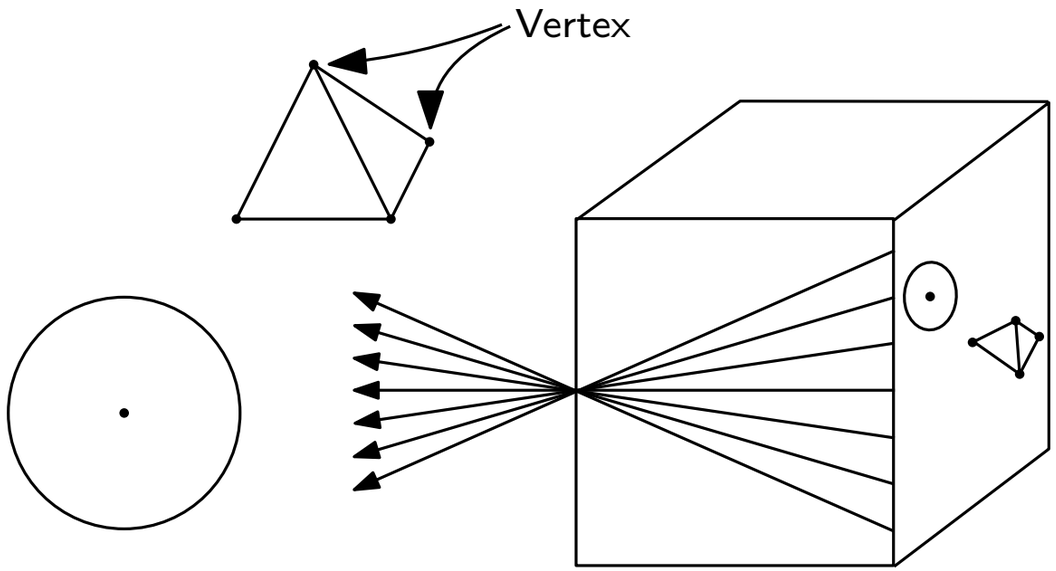


2) Rasterization

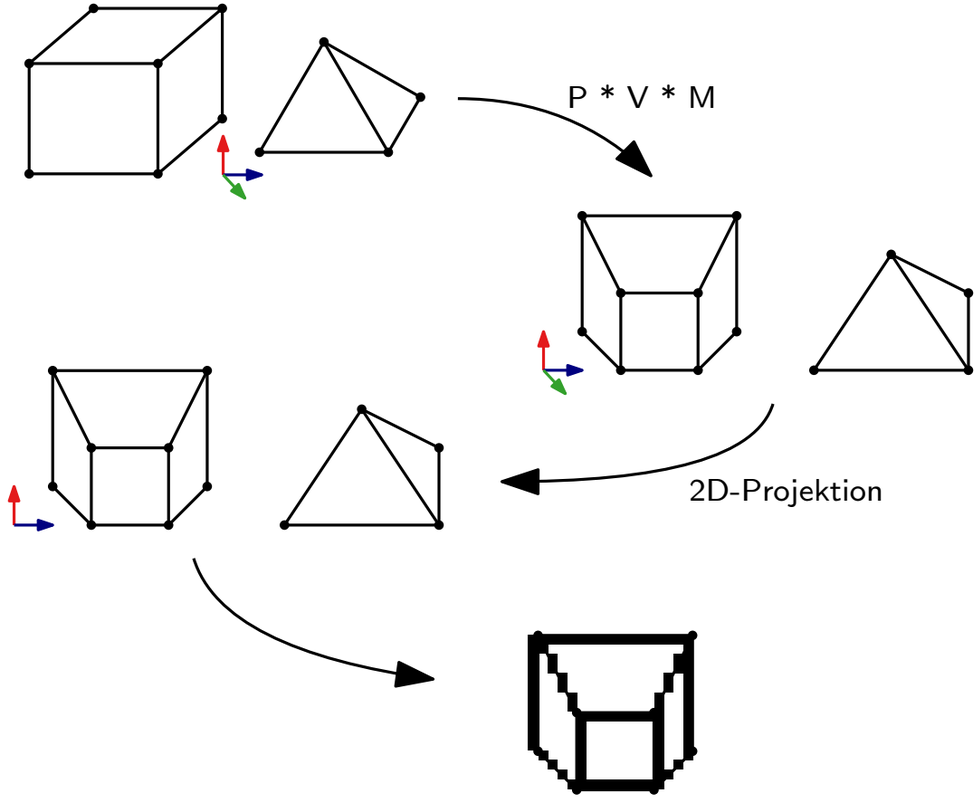


Traditionelle Rendering-Techniken

1) Raytracing



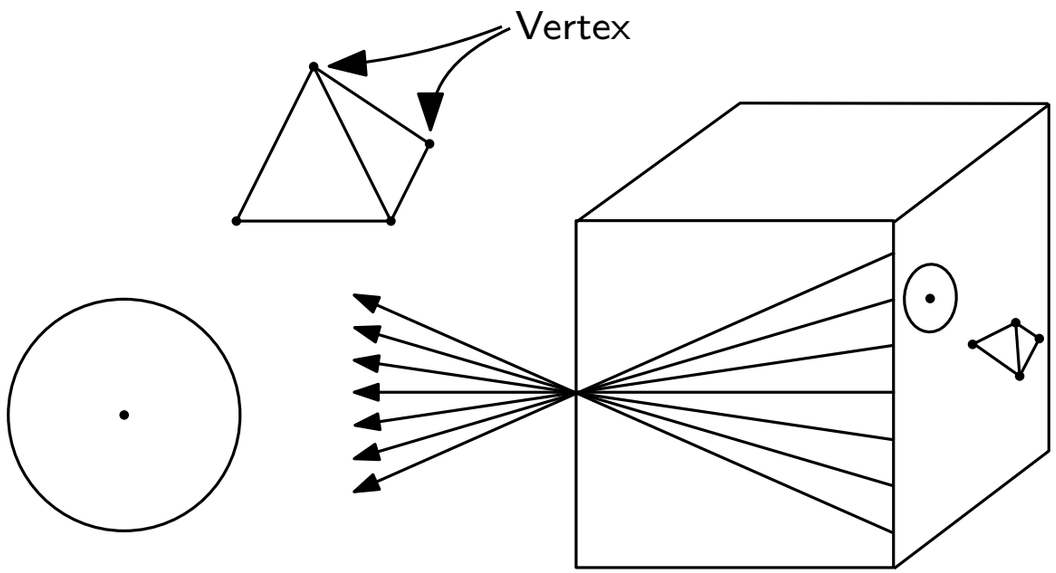
2) Rasterization



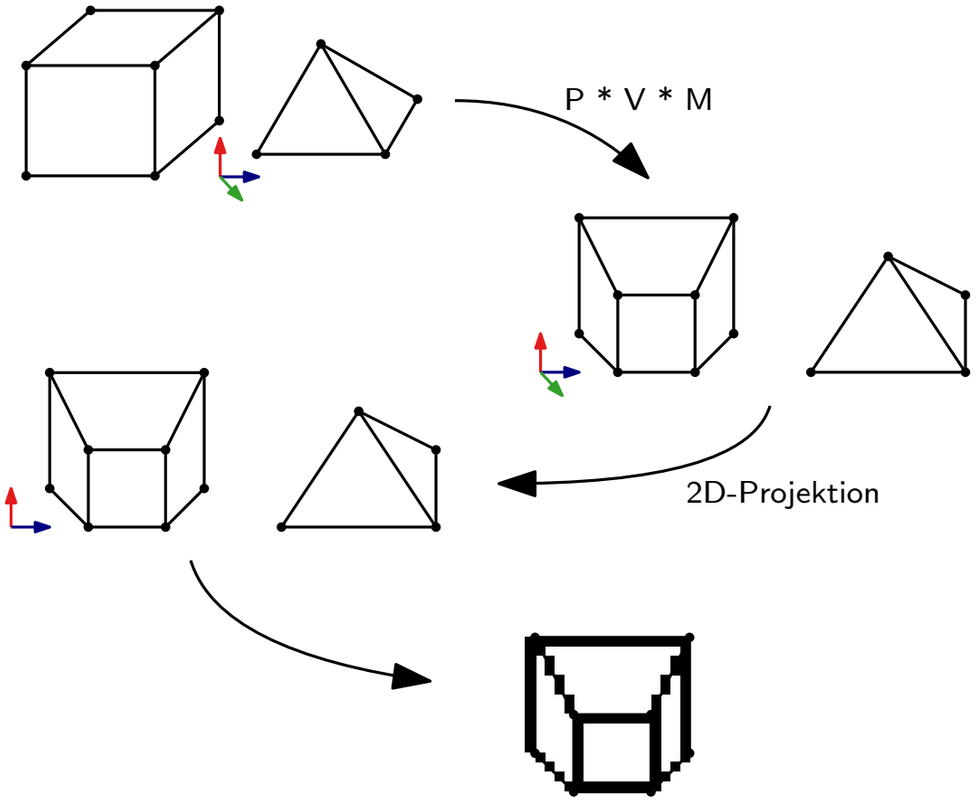
Nur die Außenhülle der Geometrie wird verwendet!

Traditionelle Rendering-Techniken

1) Raytracing



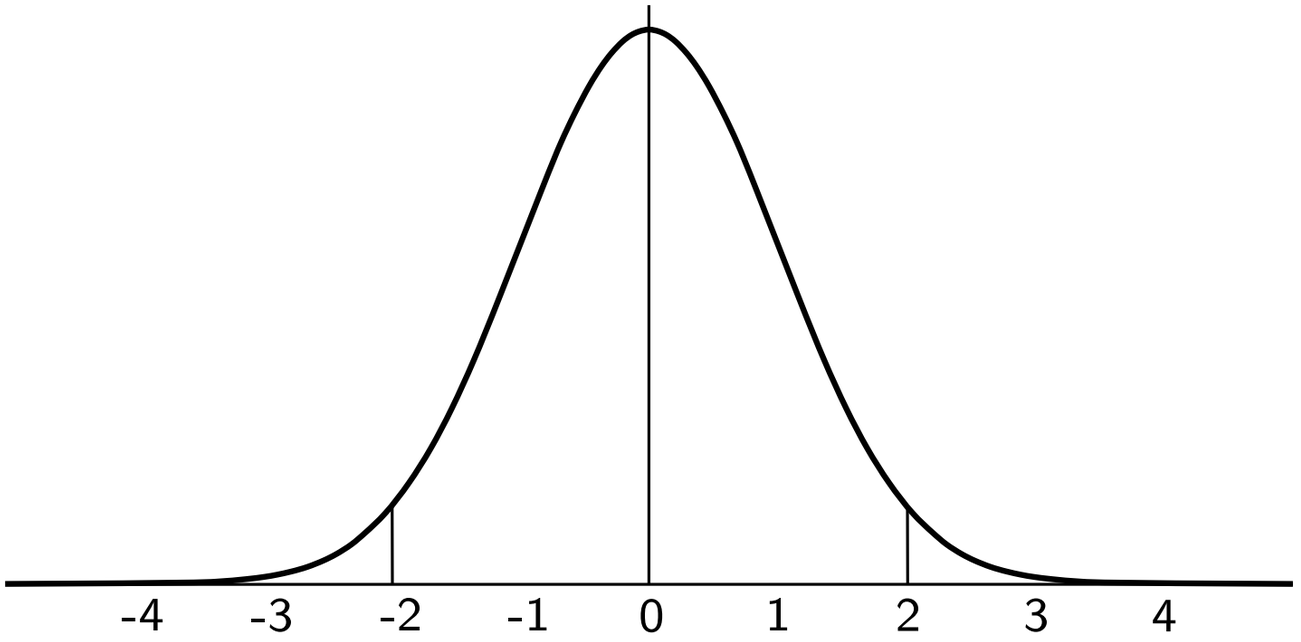
2) Rasterization



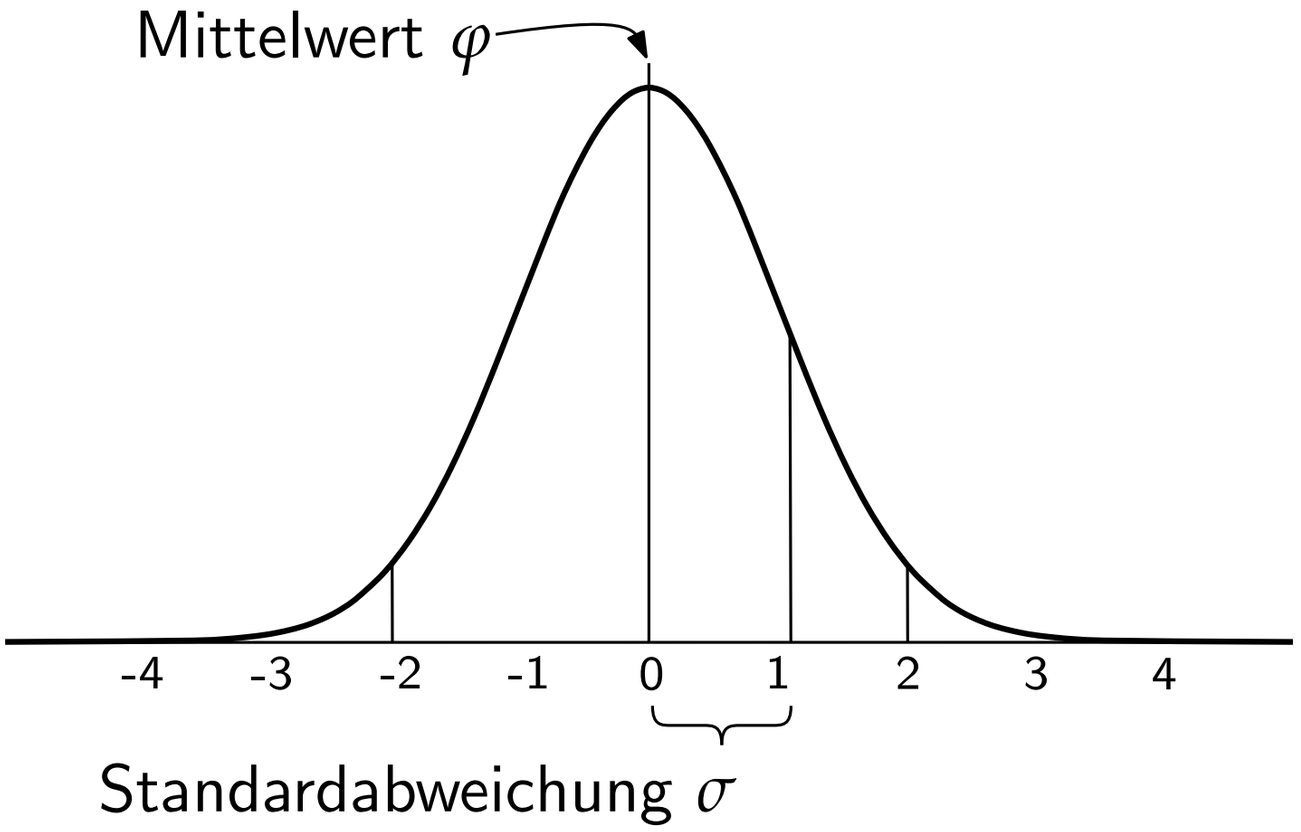
Nur die Außenhülle der Geometrie wird verwendet!

3D-Gaussian Splatting $\Rightarrow$ Objekte werden als Volumen repräsentiert!

Was war nochmal Normalverteilung?

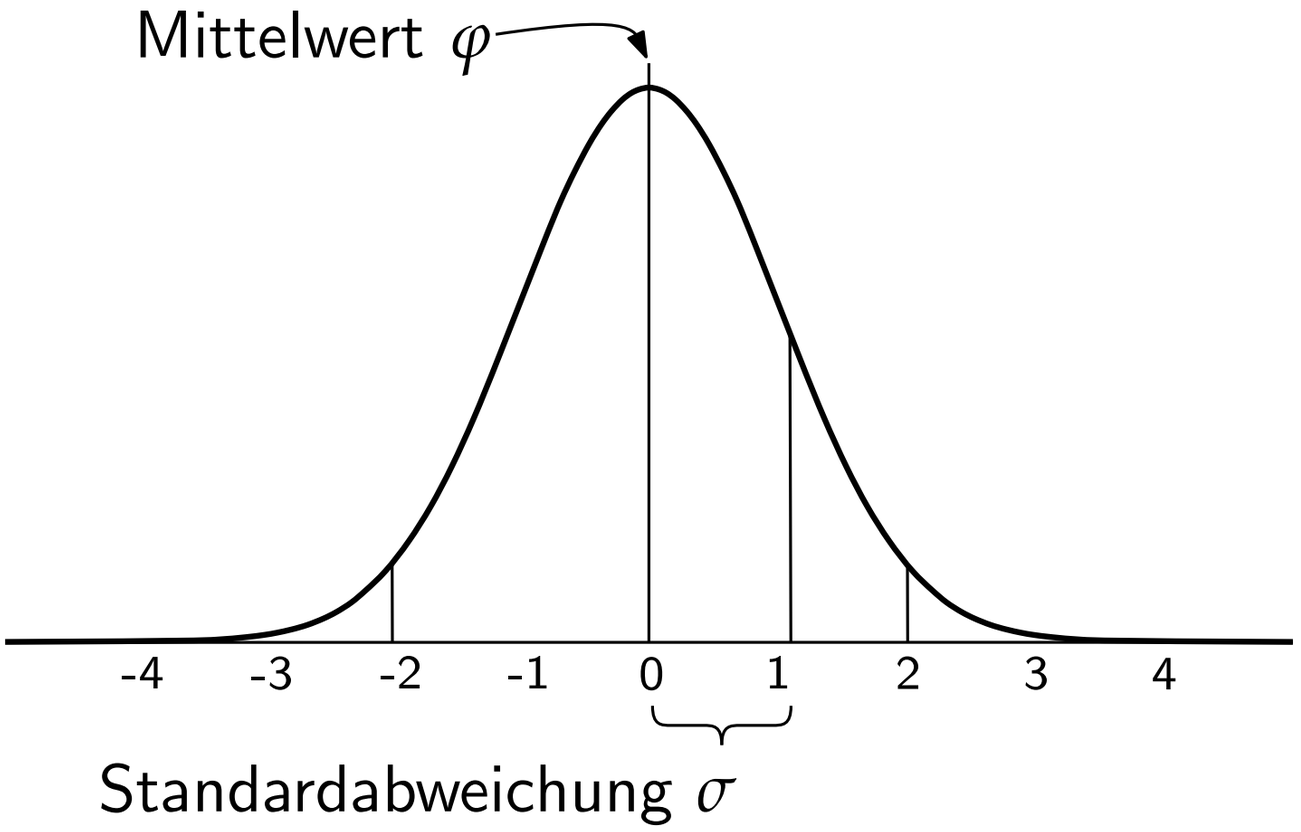


Was war nochmal Normalverteilung?



Was war nochmal Normalverteilung?

$$f(x|\varphi,\sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\varphi}{\sigma}\right)^2}$$

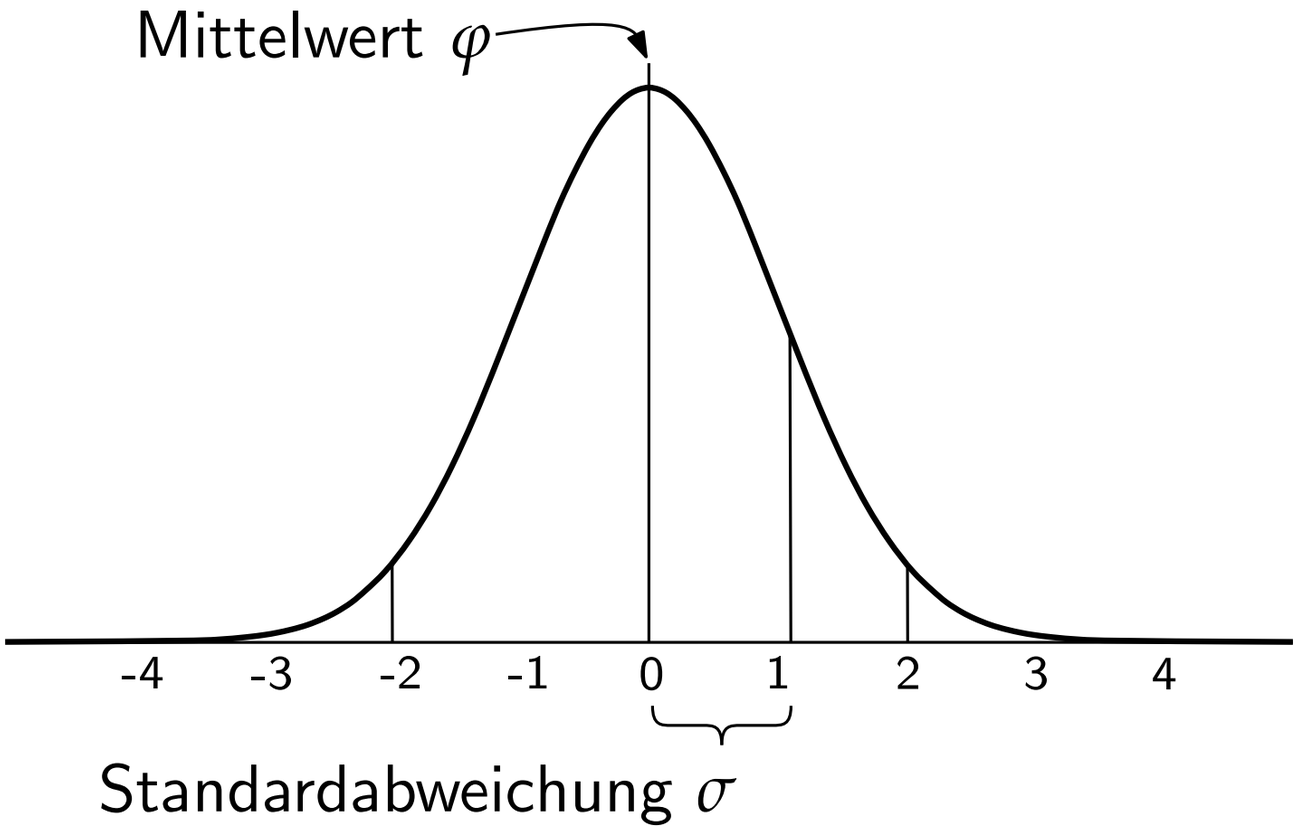


Was war nochmal Normalverteilung?

(Demo 2)

Was war nochmal Normalverteilung?

$$f(x|\varphi,\sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\varphi}{\sigma}\right)^2}$$



2 3D Gaussian

1D: $$f(x|\varphi,\sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\varphi)^2}{2\sigma^2}}$$

2 3D Gaussian

1D: $$f(x|\varphi,\sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\varphi)^2}{2\sigma^2}}$$

2D: $$f(x,y|\varphi_x,\varphi_y,\sigma_x^2,\sigma_y^2) = Ae^{-\left(\frac{(x-\varphi_x)^2}{2\sigma_x^2}+\frac{(y-\varphi_y)^2}{2\sigma_y^2}\right)}$$

2 3D Gaussian

1D: $$f(x|\varphi,\sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\varphi)^2}{2\sigma^2}}$$

2D: $$f(x,y|\varphi_x,\varphi_y,\sigma_x^2,\sigma_y^2) = Ae^{-\left(\frac{(x-\varphi_x)^2}{2\sigma_x^2}+\frac{(y-\varphi_y)^2}{2\sigma_y^2}\right)}$$

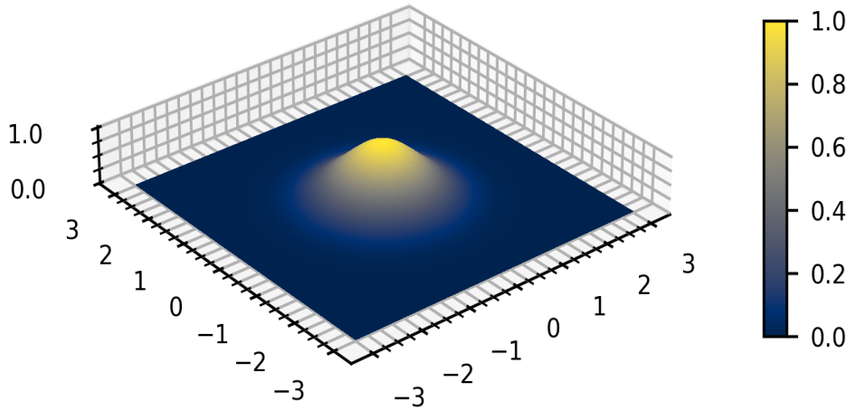
(Demo 3)

2 3D Gaussian

1D: $$f(x|\varphi,\sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\varphi)^2}{2\sigma^2}}$$

$$\downarrow$$

2D: $$f(x,y|\varphi_x,\varphi_y,\sigma_x^2,\sigma_y^2) = Ae^{-\left(\frac{(x-\varphi_x)^2}{2\sigma_x^2}+\frac{(y-\varphi_y)^2}{2\sigma_y^2}\right)}$$



Quelle Bild: https://en.wikipedia.org/wiki/Gaussian_function

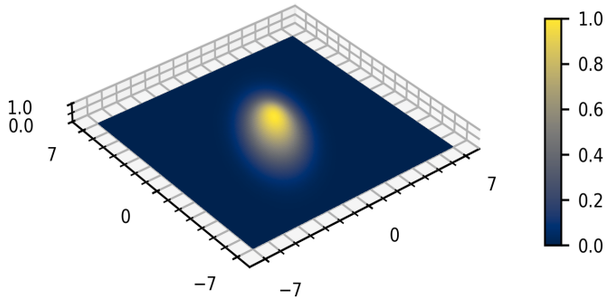
2 ~~3~~D Gaussian

1D: $$f(x|\varphi,\sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\varphi)^2}{2\sigma^2}}$$

$$\downarrow$$

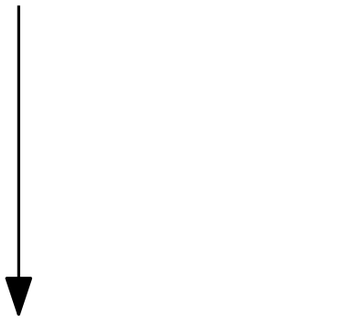
2D: $$f(x,y|\varphi_x,\varphi_y,\sigma_x^2,\sigma_y^2) = Ae^{-\left(\frac{(x-\varphi_x)^2}{2\sigma_x^2}+\frac{(y-\varphi_y)^2}{2\sigma_y^2}\right)}$$

Alternativ: $$f(x,y) = Ae^{-(a(x-x_0)^2+2b(x-x_0)(y-y_0)+c(y-y_0)^2)}$$

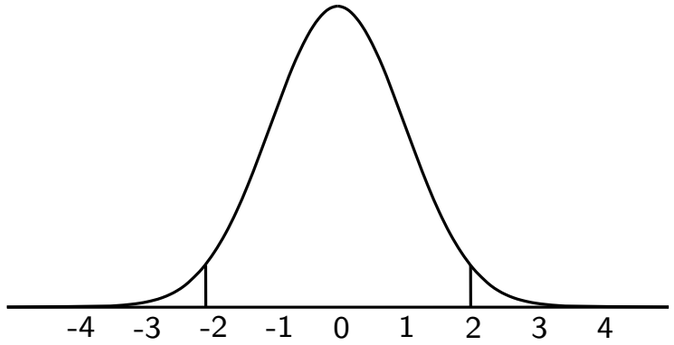


3D Gaussian?

1D-Gaussian

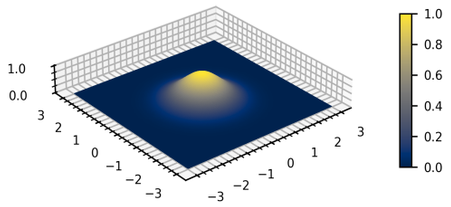


2D-Projektion



2D-Gaussian

3D-Projektion

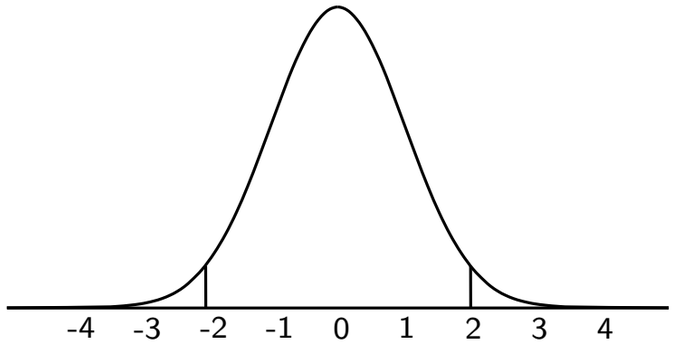


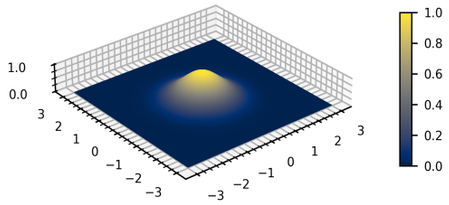
3D-Gaussian

4D-Projektion

??

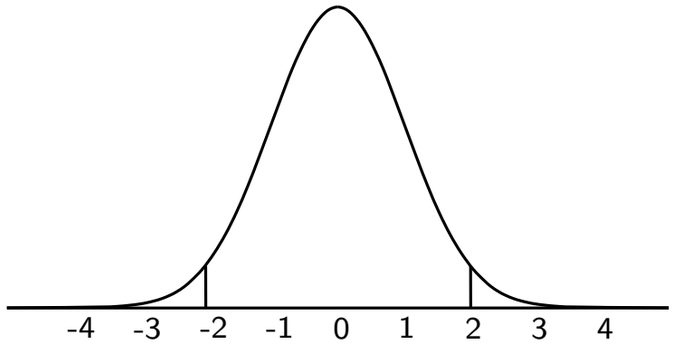
Ein anderer Ansatz...

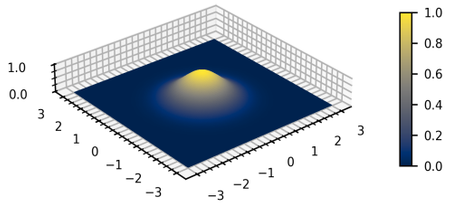




3D-Gaussian

Ein anderer Ansatz...

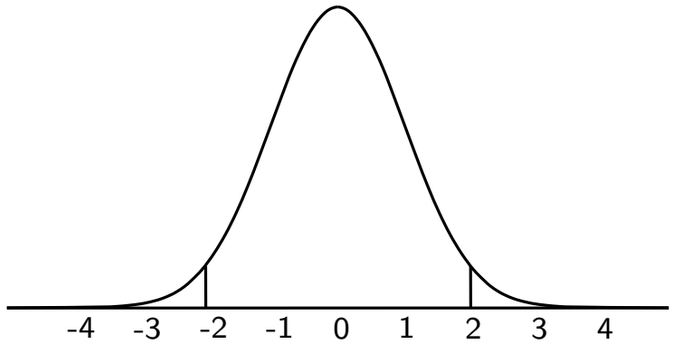




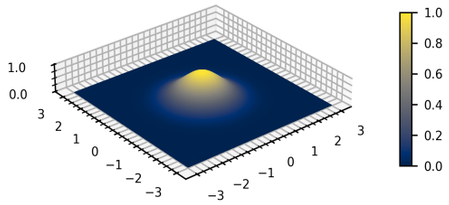
3D-Gaussian

(Demo)

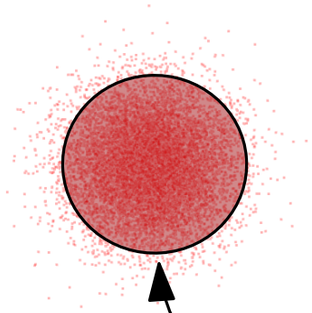
Ein anderer Ansatz...



Konfidenzinterval



3D-Gaussian



Konfidenzinterval
(Ellipse)

(Demo)

Konfidenzinterval
(Ellipsoid)

Überblick

Bilder von einem echten Objekt werden aufgenommen

↓

Mit Gaussians, wird die Szene digital rekonstruiert

↓

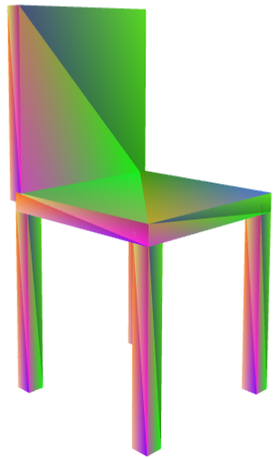
Die Szene kann aus beliebigen Blickwinkeln betrachtet werden

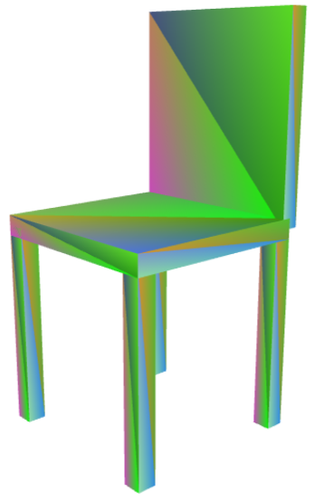
DEMO

Von Bildern zu 3D-Gaussians

Der Plan:

Input: Bilder und Point Cloud von statischer Szene aus verschiedenen Perspektiven





Von Bildern zu 3D-Gaussians

Der Plan:

Input: Bilder und Point Cloud von statischer Szene aus verschiedenen Perspektiven

- Szene mit 3D-Gaussians rekonstruieren = Training
- Gegebene Gaussians Zeichnen = Rendern

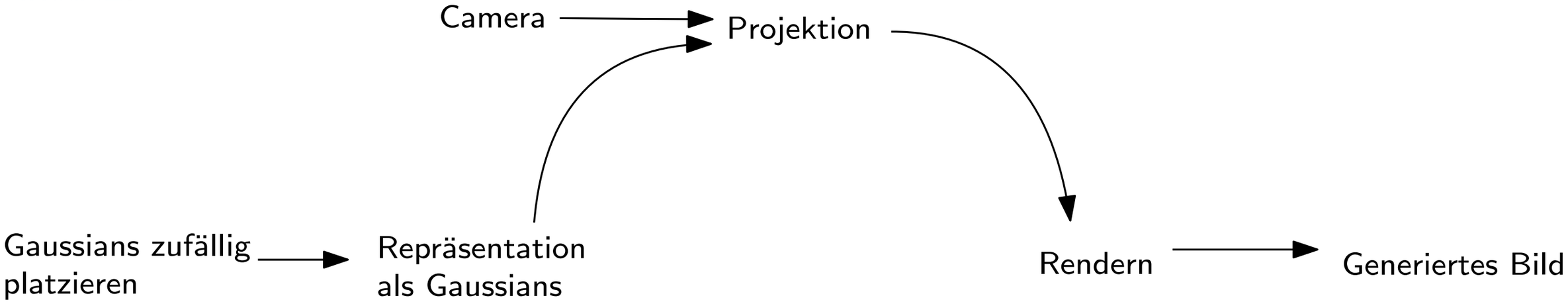
Von Bildern zu 3D-Gaussians

Der Plan:

Input: Bilder und Point Cloud von statischer Szene aus verschiedenen Perspektiven

- Szene mit 3D-Gaussians rekonstruieren = Training

Ansatz:



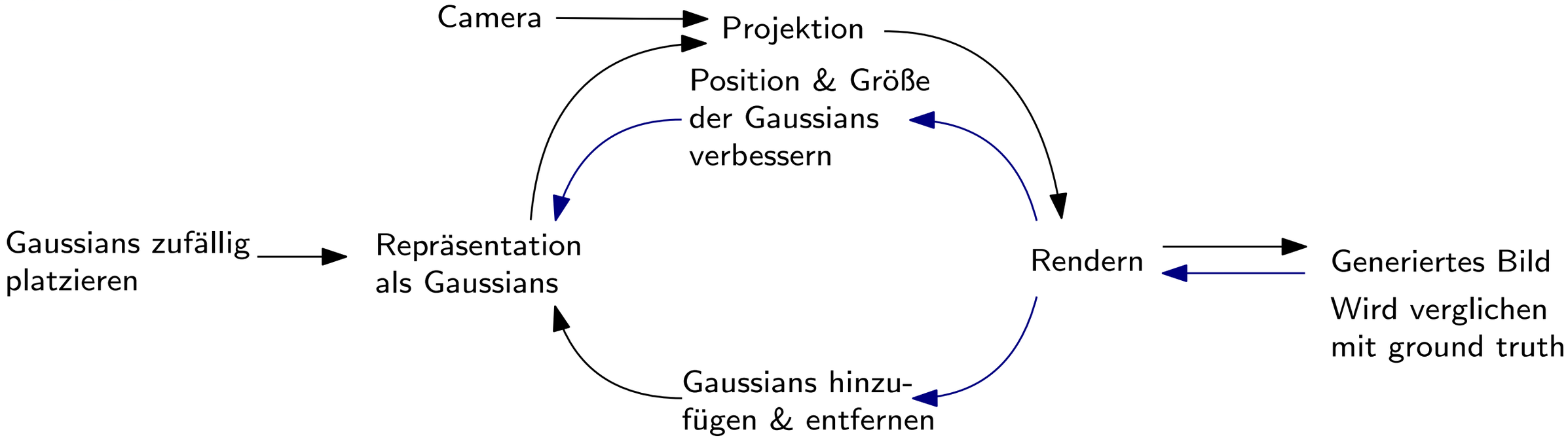
Von Bildern zu 3D-Gaussians

Der Plan:

Input: Bilder und Point Cloud von statischer Szene aus verschiedenen Perspektiven

- Szene mit 3D-Gaussians rekonstruieren = Training

Ansatz:



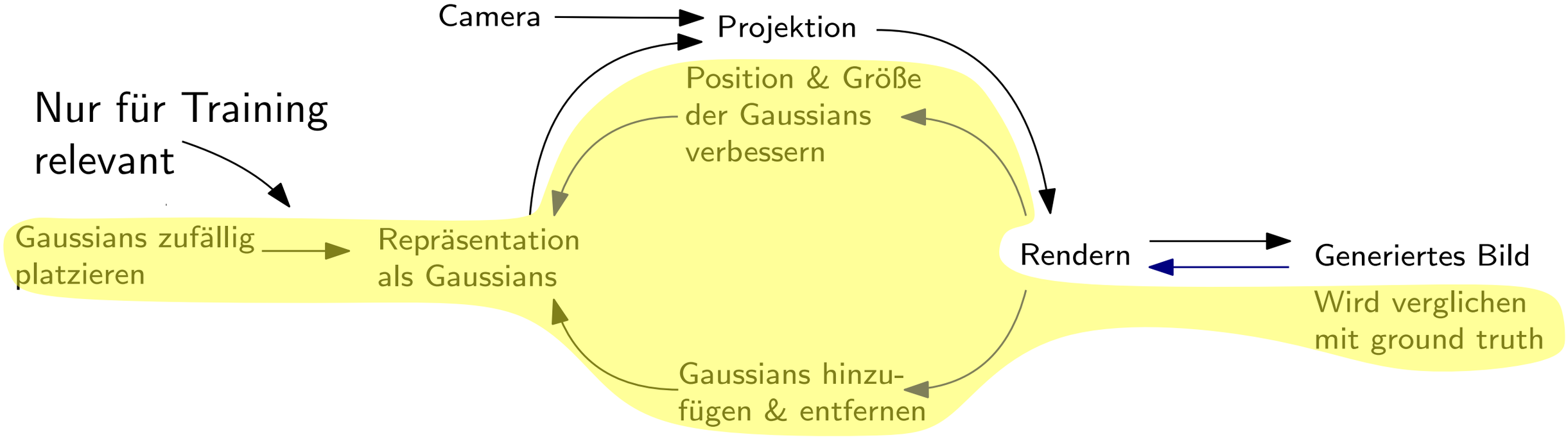
Von Bildern zu 3D-Gaussians

Der Plan:

Input: Bilder und Point Cloud von statischer Szene aus verschiedenen Perspektiven

- Szene mit 3D-Gaussians rekonstruieren = Training

Ansatz:



Szene rekonstruieren

Nach welchen Kriterien werden Gaussians verschoben, erstellt und gelöscht?

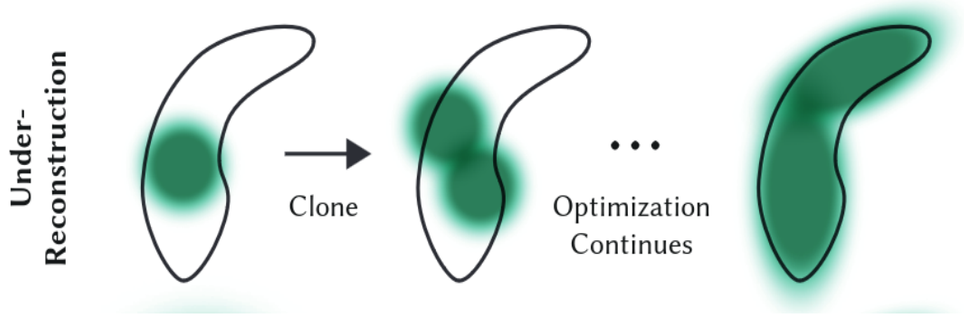
- Start mit wenigen isotropischen Gaussians
- Jede Iteration werden Position, Größe, Farbe und Transparenz optimiert.

Alle 100 Iterationen:

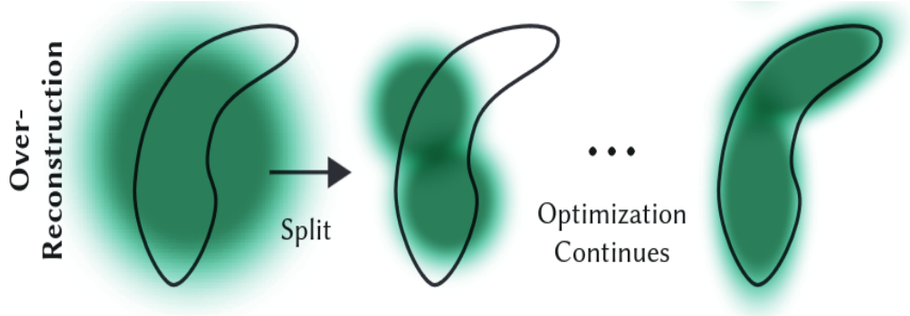
- Sehr große oder sehr transparente Gaussians ($\alpha < \varepsilon_\alpha$) werden entfernt

Wenn zu wenige oder zu große Gaussians vorliegen:

Under-reconstruction



Over-reconstruction



Training - Pseudocode

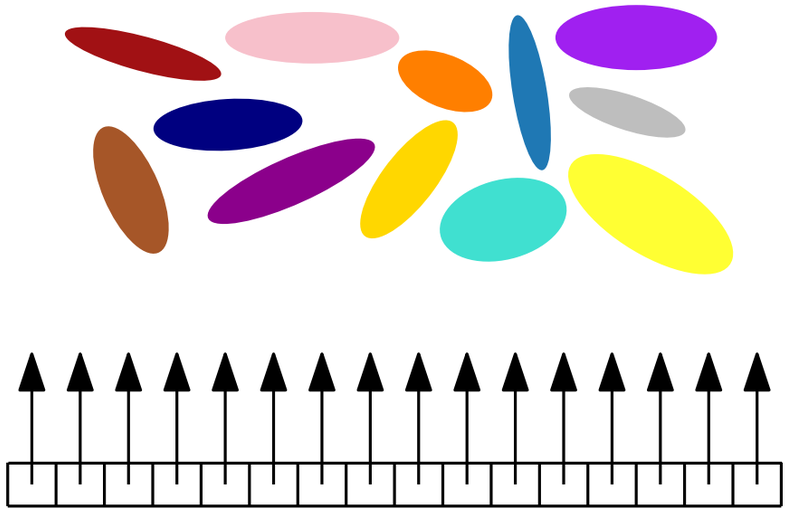
w, h: width and height of the training images

$M \leftarrow$ SfM Points ▷ Positions
$S, C, A \leftarrow$ InitAttributes() ▷ Covariances, Colors, Opacities
$i \leftarrow 0$ ▷ Iteration Count
while not converged **do**
  $V, \hat{I} \leftarrow$ SampleTrainingView() ▷ Camera V and Image
  $I \leftarrow$ Rasterize(M, S, C, A, V) ▷ Alg. 2
  $L \leftarrow Loss(I, \hat{I})$ ▷ Loss
  $M, S, C, A \leftarrow$ Adam(∇L) ▷ Backprop & Step
  if IsRefinementIteration(i) **then**
    for all Gaussians (μ, Σ, c, α) **in** (M, S, C, A) **do**
      if $\alpha < \epsilon$ or IsTooLarge(μ, Σ) **then** ▷ Pruning
        RemoveGaussian()
      end if
      if $\nabla_p L > \tau_p$ **then** ▷ Densification
        if $\|S\| > \tau_S$ **then** ▷ Over-reconstruction
          SplitGaussian(μ, Σ, c, α)
        else ▷ Under-reconstruction
          CloneGaussian(μ, Σ, c, α)
        end if
      end if
    end for
  end if
  $i \leftarrow i + 1$
end while

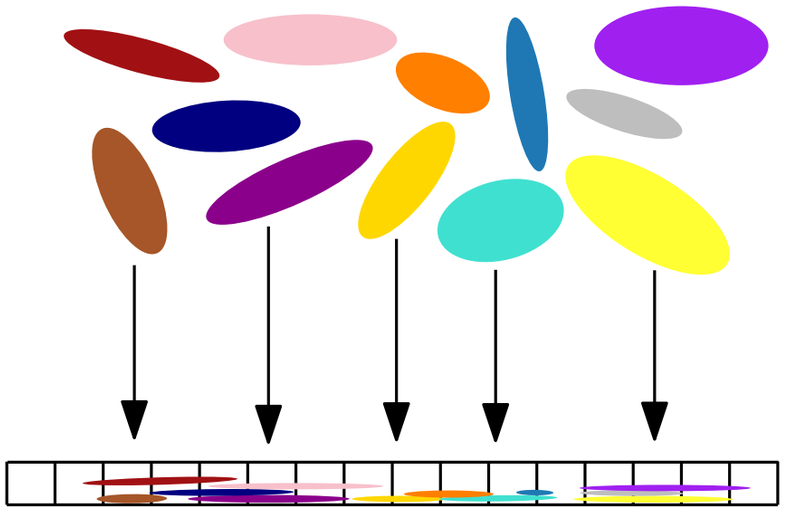
Rendering

Wie kommt das 3D-gaussian in den Pixel?

Option 1: Raytracing/Raymarching



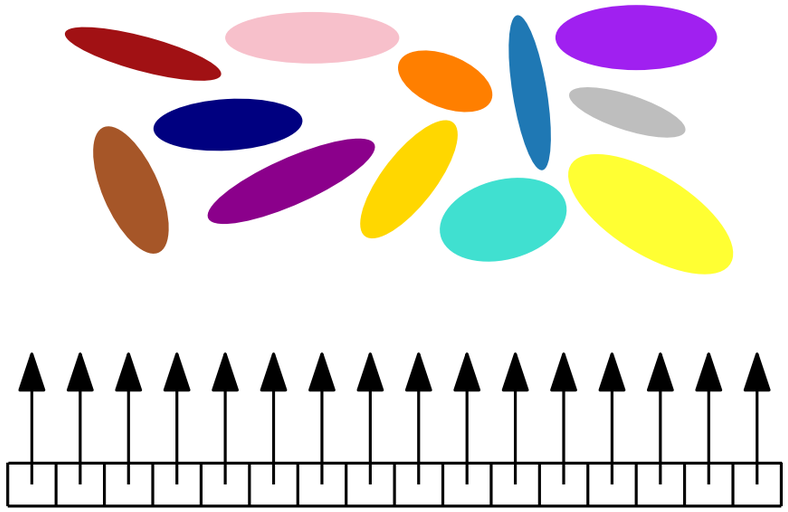
Option 2: Projektion der Gaussians auf 2D-Bildschirm



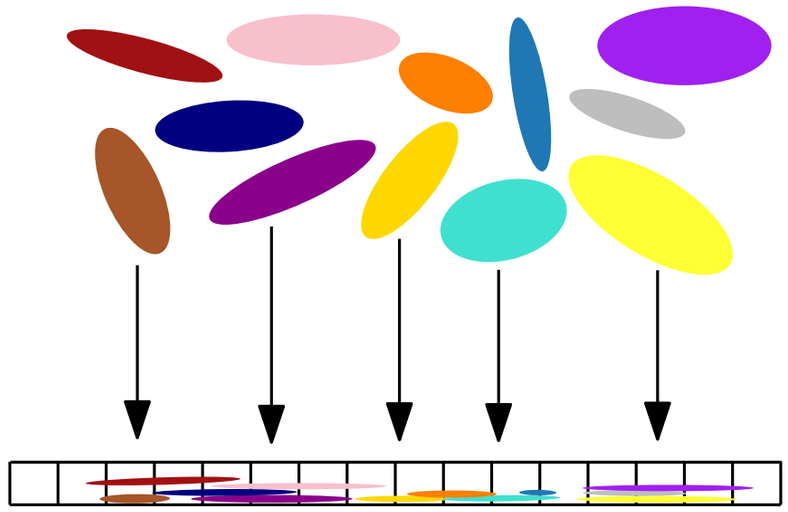
Rendering

Wie kommt das 3D-gaussian in den Pixel?

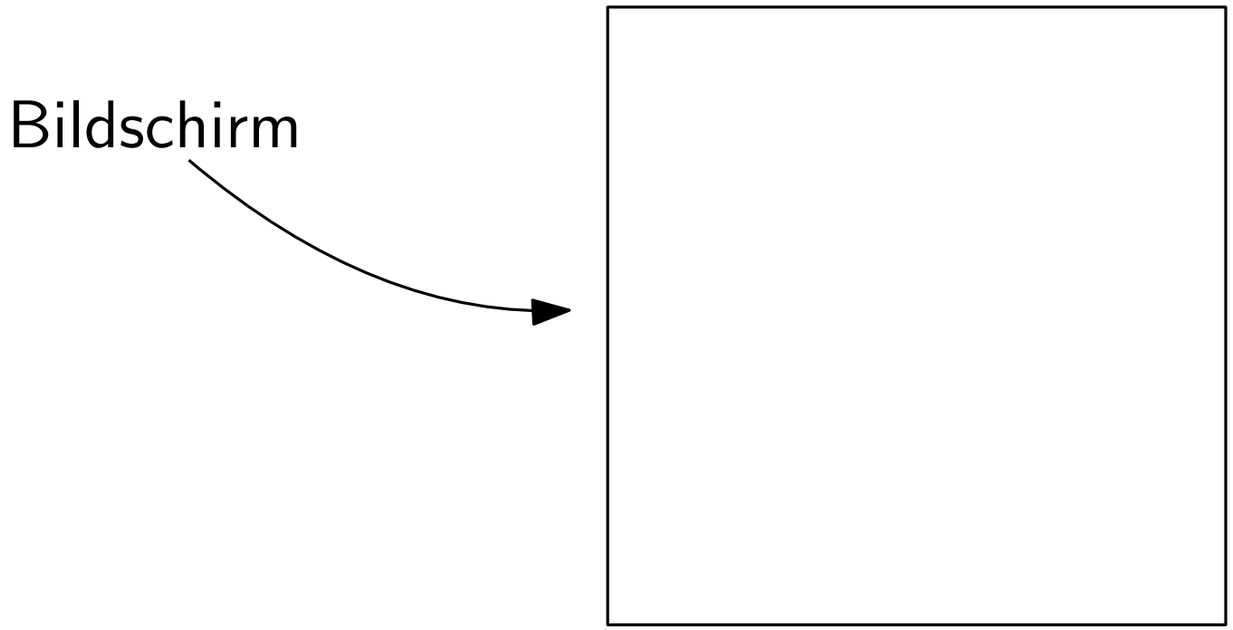
Option 1: Raytracing/Raymarching ✗



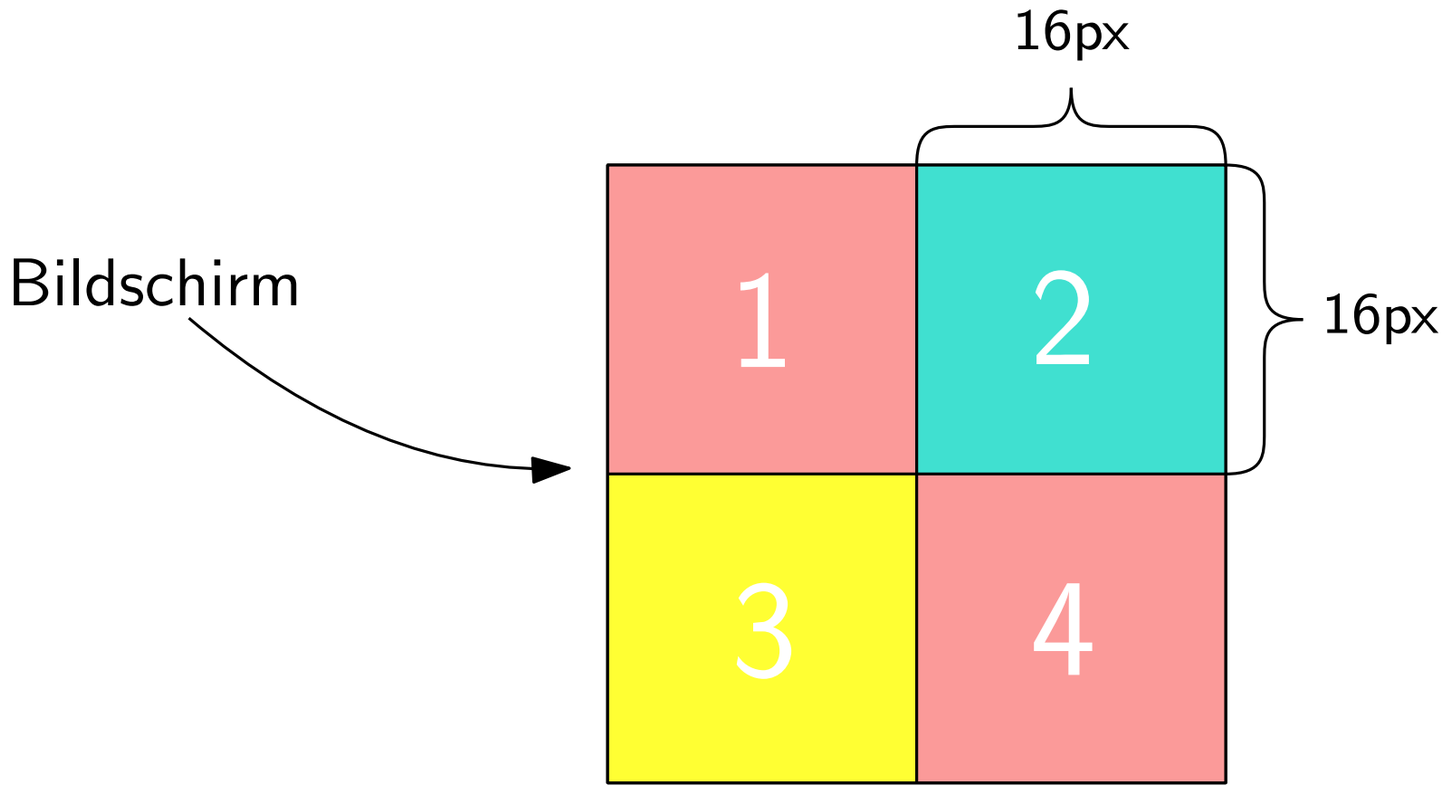
Option 2: Projektion der Gaussians auf 2D-Bildschirm ✓



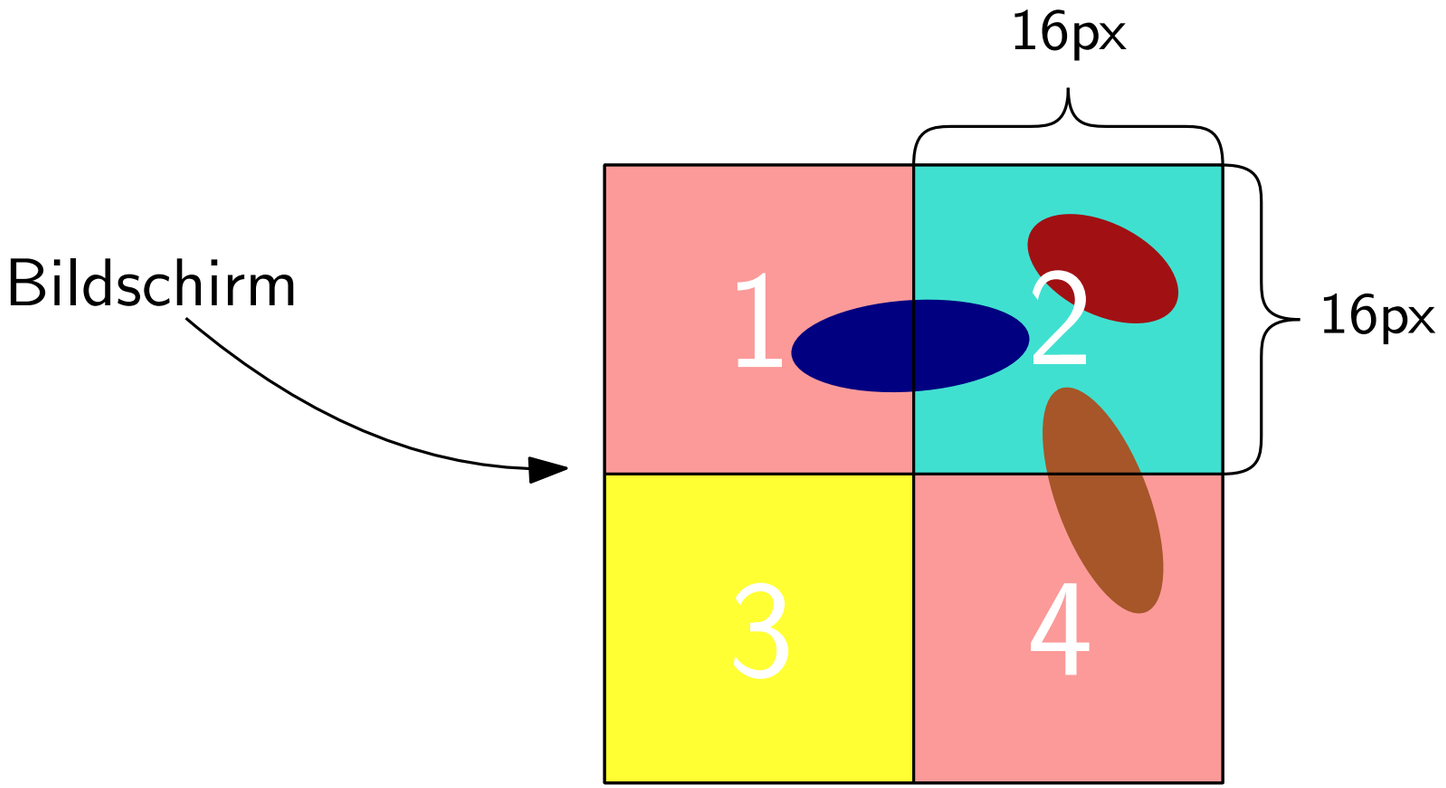
Rendering



Rendering

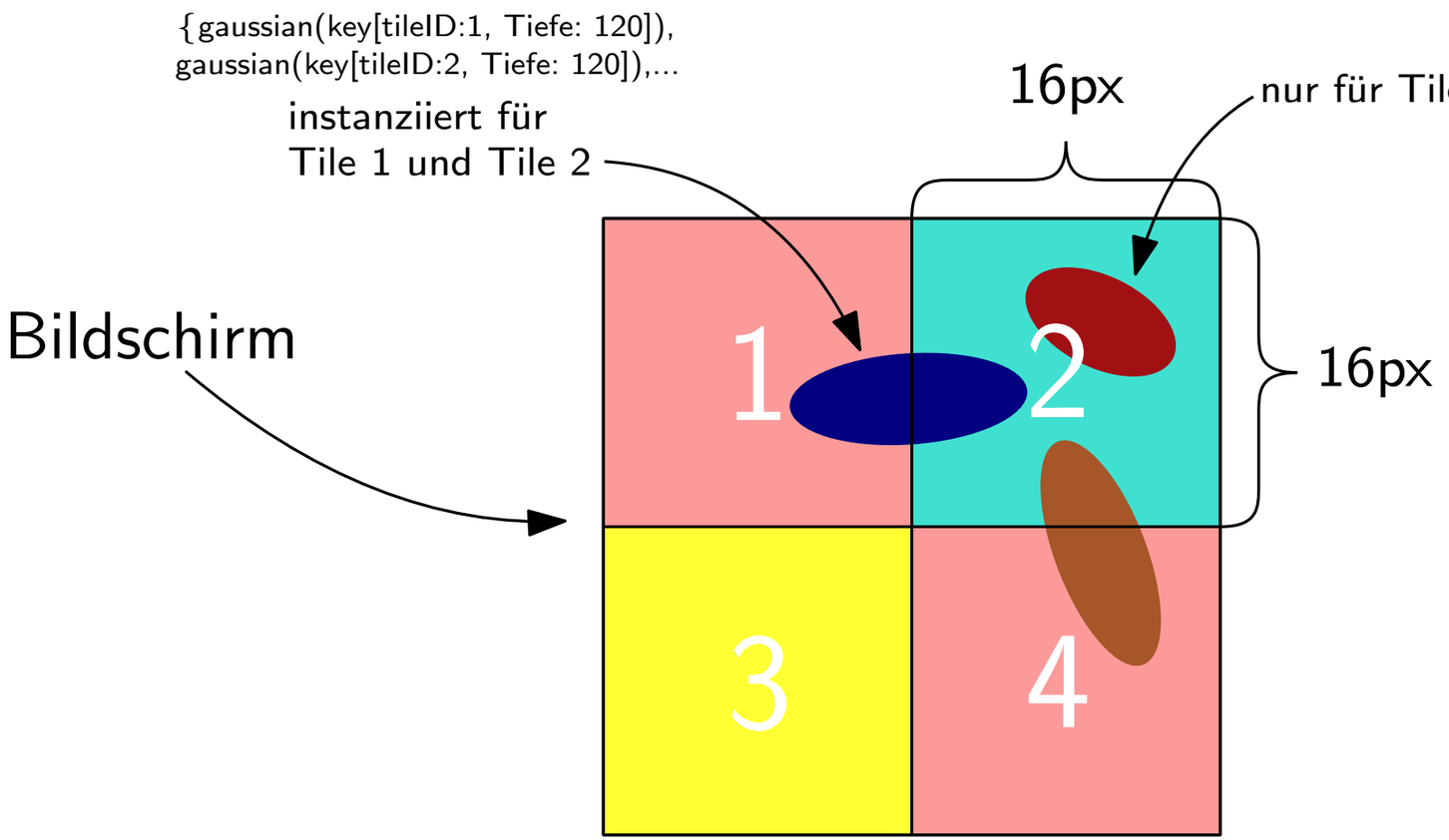


Rendering



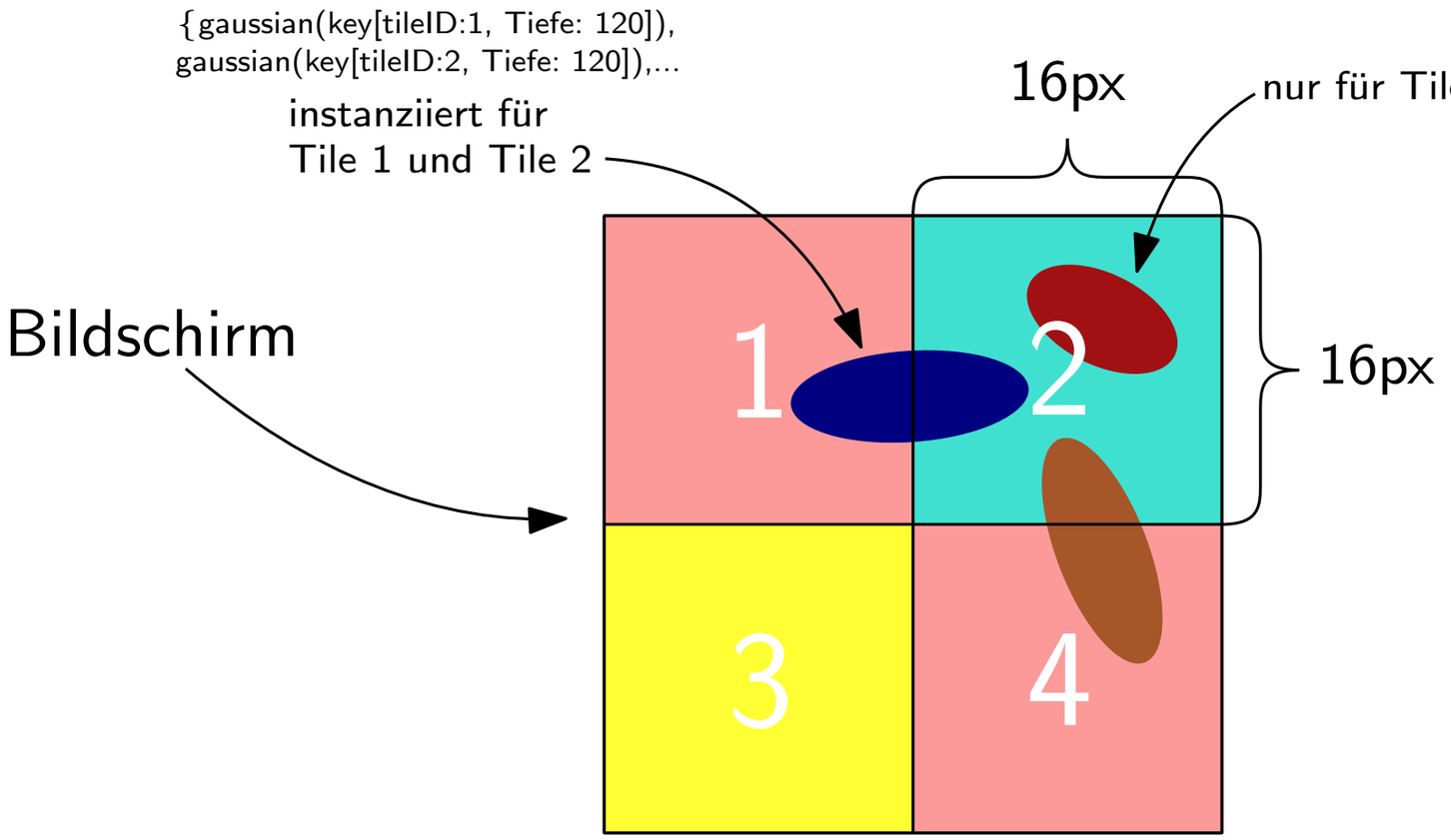
- Gaussians werden mit View-Frustum geculled (Kondidenz-interval)

Rendering



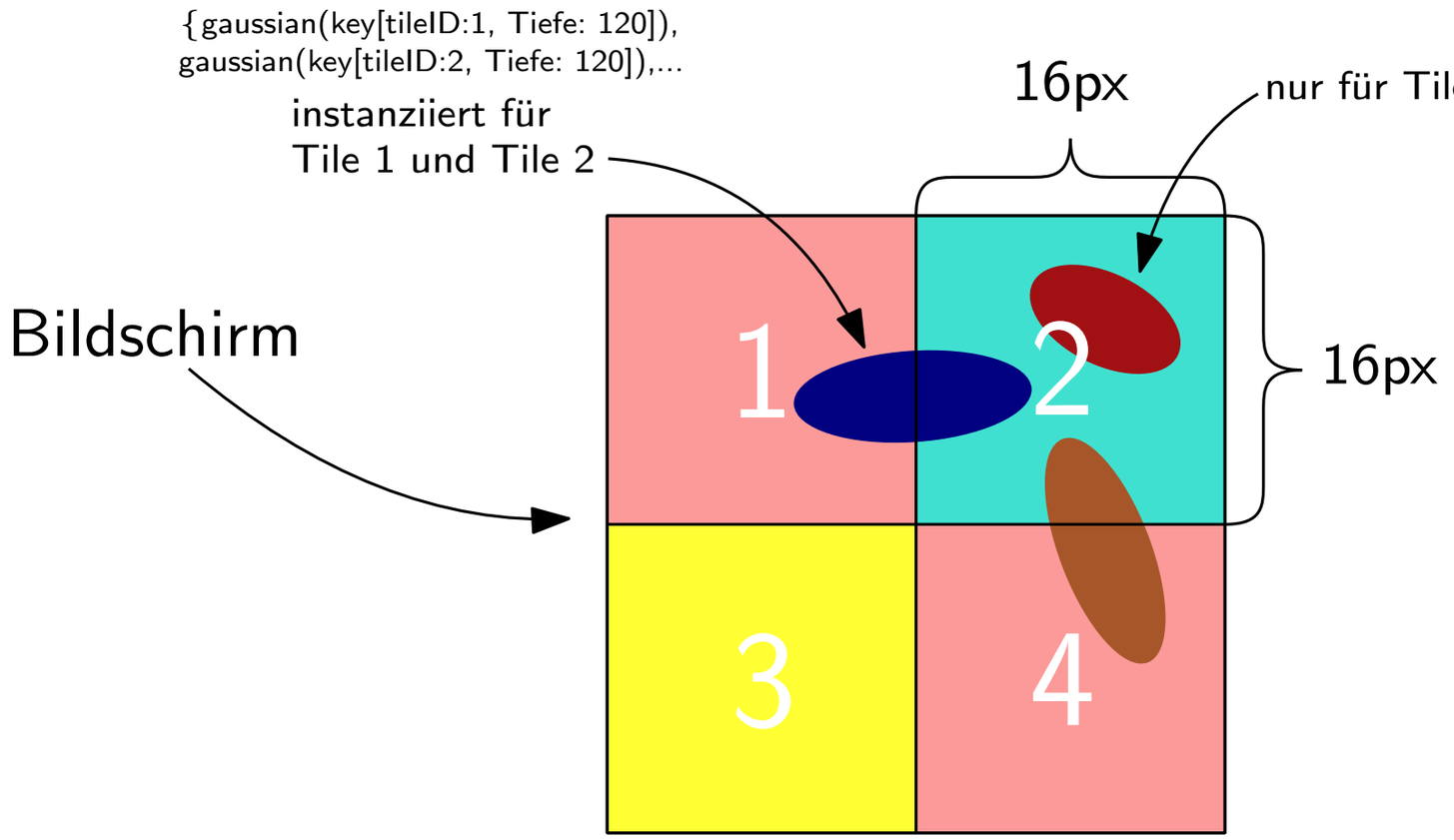
- Gaussians werden mit View-Frustum geculled (Kondidenz-interval)

Rendering



- Gaussians werden mit View-Frustum geculled (Kondidenz-interval)
- Jedes Gaussian wird pro Tile Instanziiert
- Jedes Gaussian erhält key zusammengesetzt aus TileID und view-space Tiefe

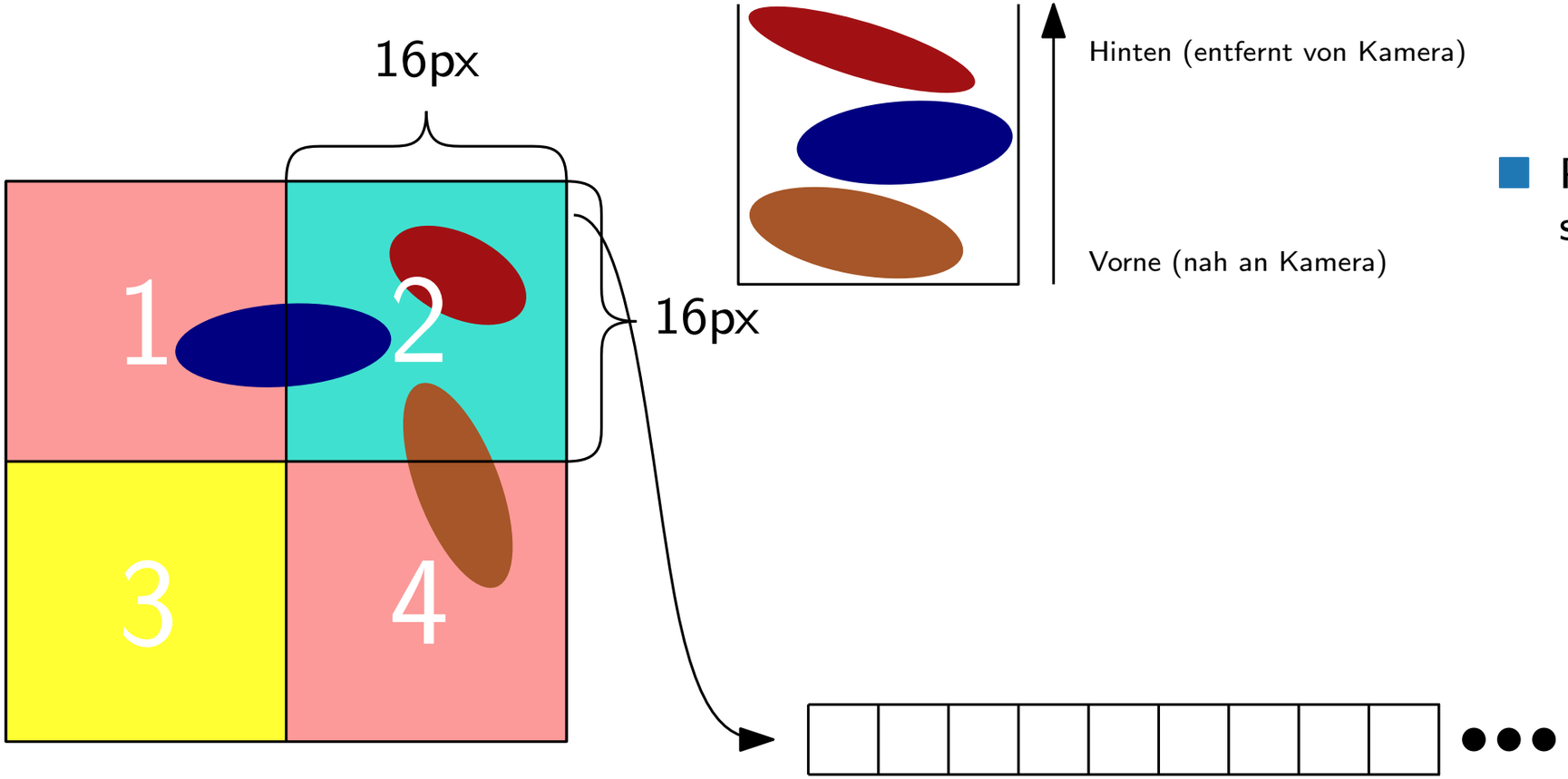
Rendering



- Gaussians werden mit View-Frustum geculled (Kondidenz-interval)
- Jedes Gaussian wird pro Tile Instanziiert
- Jedes Gaussian erhält key zusammengesetzt aus TileID und view-space Tiefe
- Gaussians werden nach Keys sortiert
- Pro Tile wird sortierte Liste mit allen Gaussians erstellt (von nah nach fern)

Rendering - Forward

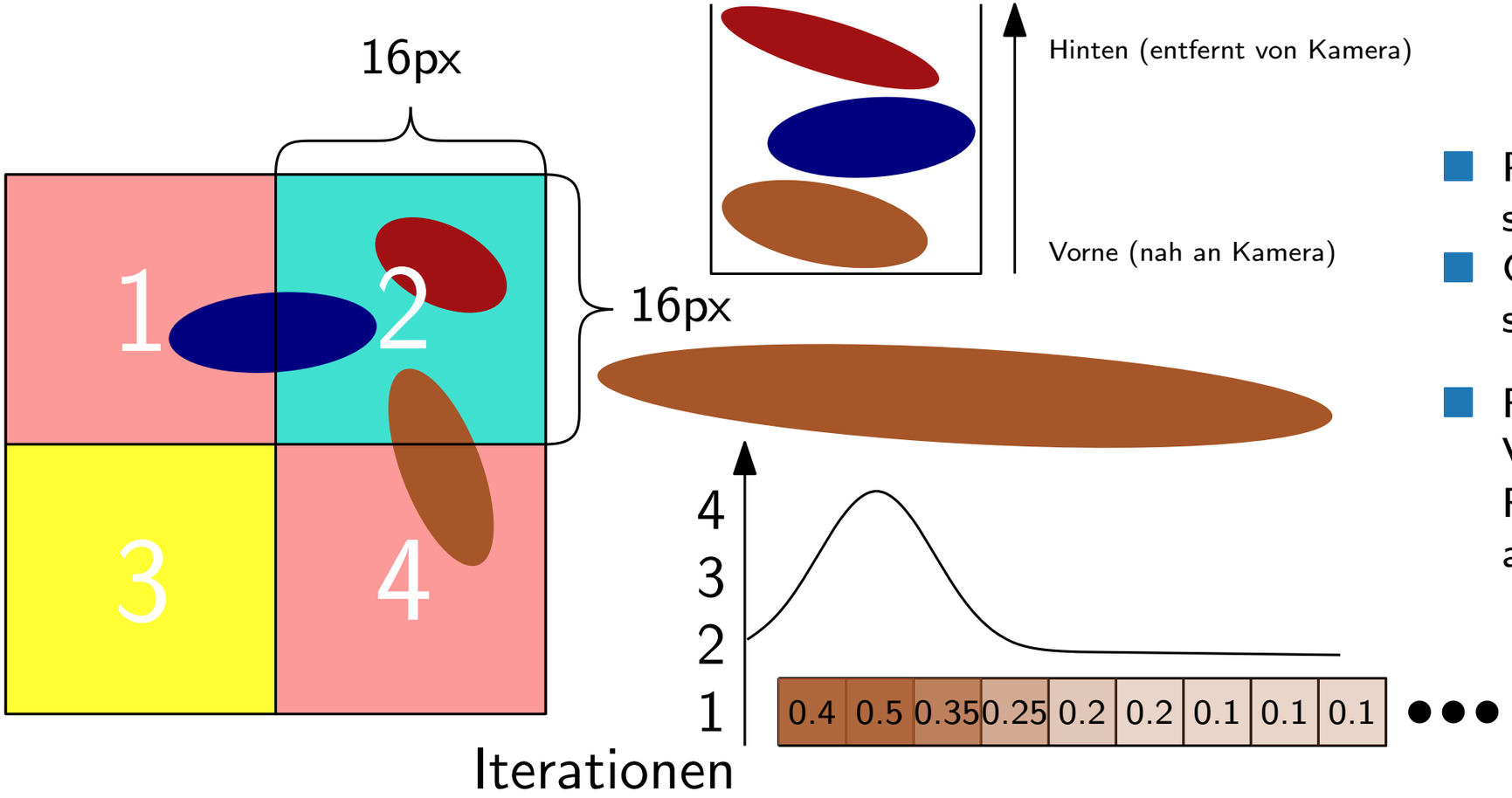
Rendering der einzelnen Tiles:



- Pro Tile liegt Liste mit sortierten Gaussians vor

Rendering - Forward

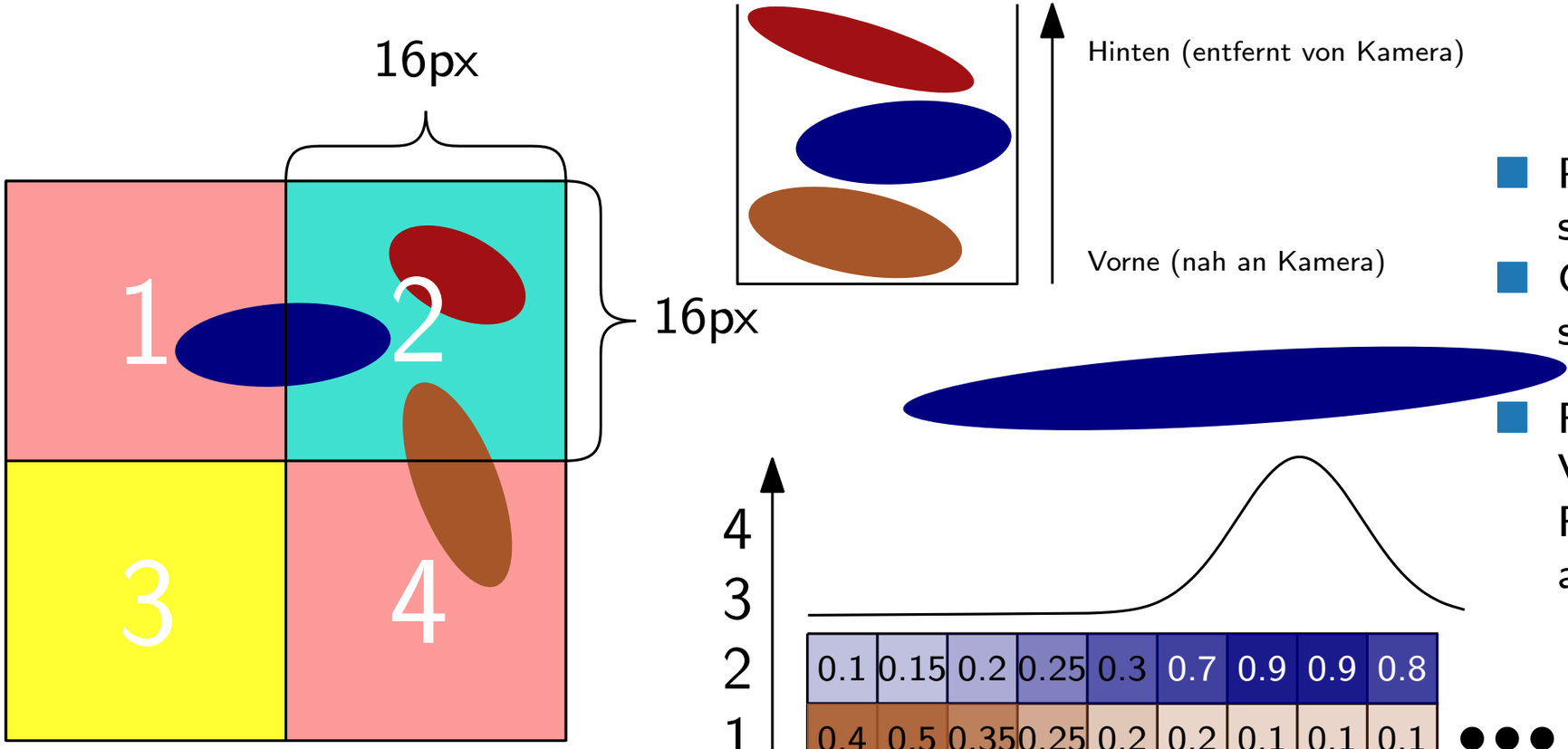
Rendering der einzelnen Tiles:



- Pro Tile liegt Liste mit sortierten Gaussians vor
- Gaussian wird 2D auf Bildschirm projeziert
- Farbwert wird entsprechend Verteilungsfkt., Alpha und Farbe von Gaussian auf Pixel akkumuliert

Rendering - Forward

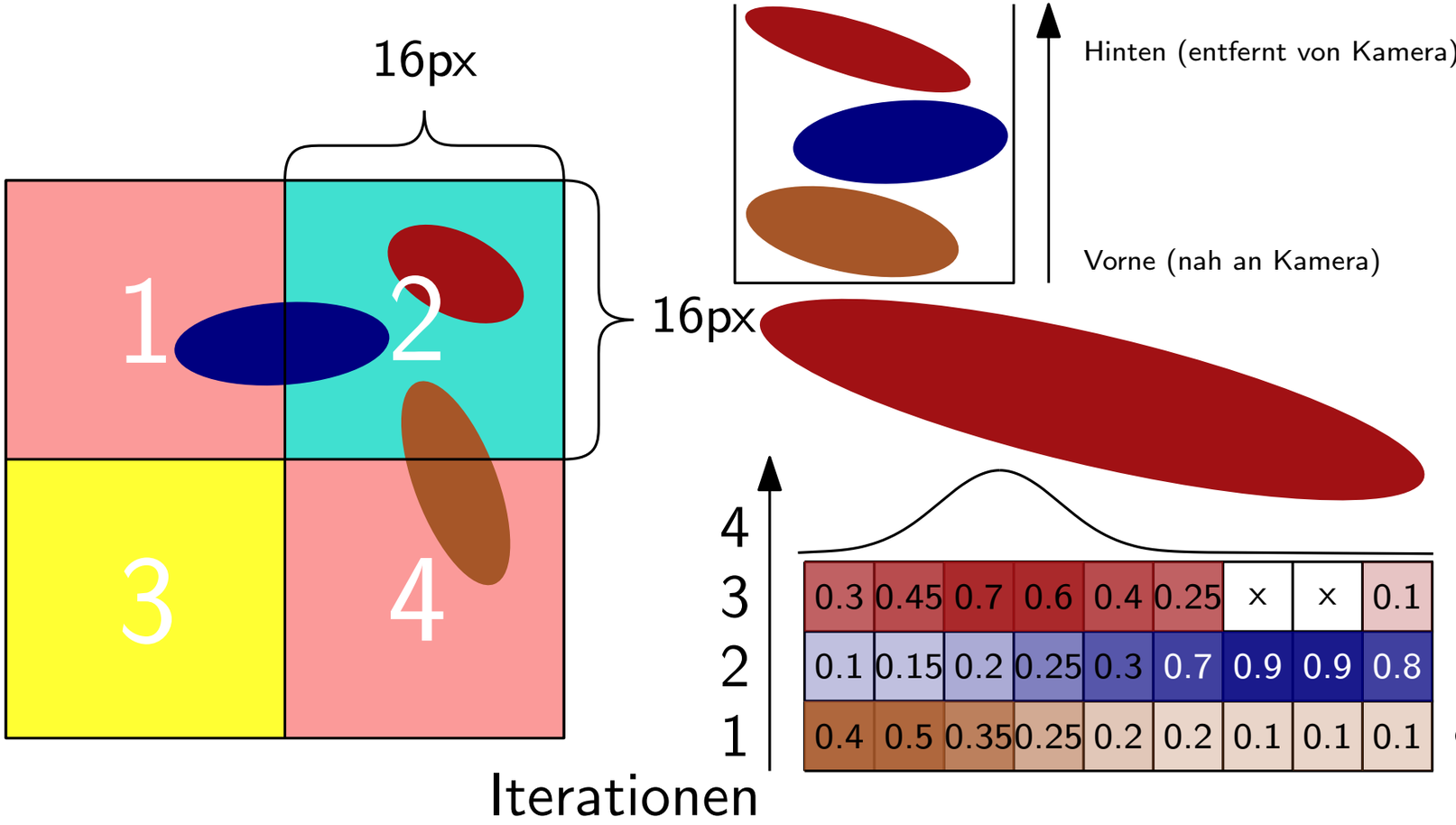
Rendering der einzelnen Tiles:



- Pro Tile liegt Liste mit sortierten Gaussians vor
- Gaussian wird 2D auf Bildschirm projeziert
- Farbwert wird entsprechend Verteilungsfkt., Alpha und Farbe von Gaussian auf Pixel akkumuliert

Rendering - Forward

Rendering der einzelnen Tiles:



- Pro Tile liegt Liste mit sortierten Gaussians vor
- Gaussian wird 2D auf Bildschirm projeziert
- Farbwert wird entsprechend Verteilungsfkt., Alpha und Farbe von Gaussian auf Pixel akkumuliert
- Pixel mit Opacity 1 werden als "Done" markiert

  ⇒ terminiert wenn Liste leer ist oder alle Pixel "Done"

Rendering - Pseudocode

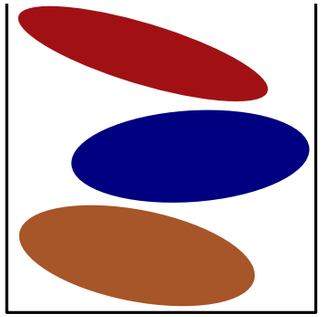
```
function RASTERIZE(w, h, M, S, C, A, V)
    CullGaussian(p, V)                          ▷ Frustum Culling
    M′, S′ ← ScreenspaceGaussians(M, S, V)      ▷ Transform
    T ← CreateTiles(w, h)
    L, K ← DuplicateWithKeys(M′, T)             ▷ Indices and Keys
    SortByKeys(K, L)                            ▷ Globally Sort
    R ← IdentifyTileRanges(T, K)
    I ← 0                                       ▷ Init Canvas
    for all Tiles t in I do
        for all Pixels i in t do
            r ← GetTileRange(R, t)
            I[i] ← BlendInOrder(i, L, r, K, M′, S′, C, A)
        end for
    end for
     return I
end function
```

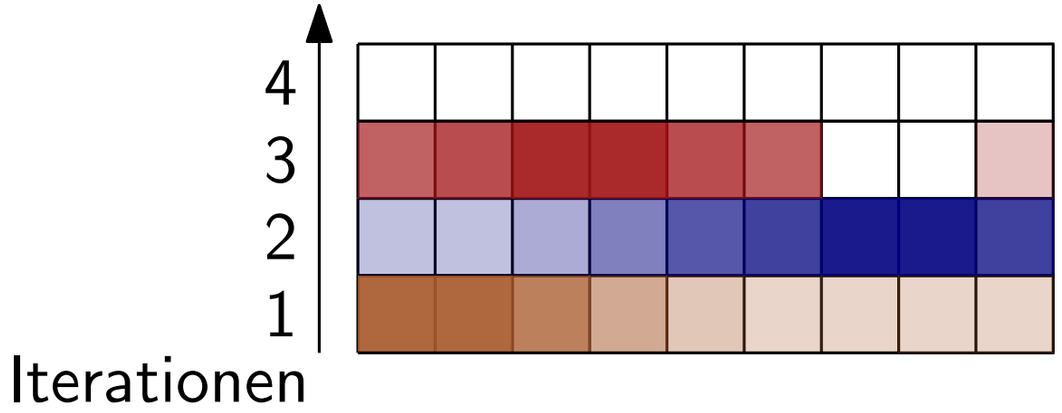
Rendering - Pseudocode

```
function RASTERIZE(w, h, M, S, C, A, V)
    CullGaussian(p, V)                          ▷ Frustum Culling
    M′, S′ ← ScreenspaceGaussians(M, S, V)      ▷ Transform
    T ← CreateTiles(w, h)
    L, K ← DuplicateWithKeys(M′, T)             ▷ Indices and Keys
    SortByKeys(K, L)                            ▷ Globally Sort
    R ← IdentifyTileRanges(T, K)
    I ← 0                                       ▷ Init Canvas
    for all Tiles t in I do
        for all Pixels i in t do
            r ← GetTileRange(R, t)
            I[i] ← BlendInOrder(i, L, r, K, M′, S′, C, A)
        end for
    end for
     return I
end function
```

Rendering - Backward

Welche Auswirkung hat ein Gaussian auf die Farbe eines Pixels?

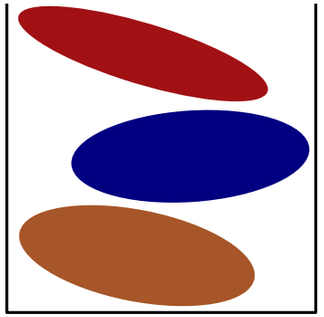




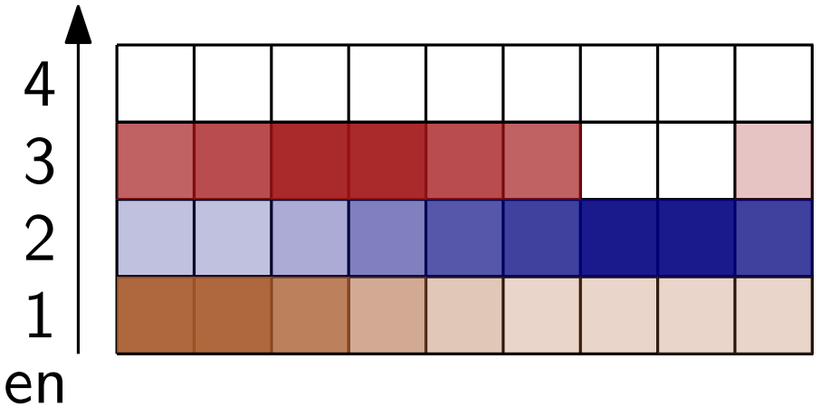
Finale Pixelfarbe:

Rendering - Backward

Welche Auswirkung hat ein Gaussian auf die Farbe eines Pixels?



- Sortierte Liste wird rückwärts durchlaufen
- Gradient Descent wird verwendet, um Optimum zu finden
- Rekonstruktion der Szene entsprechend angepasst



Finale Pixelfarbe:

Zum selbst ausprobieren:

https://gsplat.tech/

Zum selbst ausprobieren:

https://gsplat.tech/

- Viewer ohne eigenes Training
- Implementierung in WebGL
- Sortierung der Gaussians in WebAssembly

spannend und lustig...

3D Gaussian Splatting

IPE Abstürze: 5

3D Gaussian Splatting

IPE Abstürze: 5