Auto-Tuning Intermediate Representations for In Situ Visualization

Steffen Frey and Thomas Ertl

2016 New York Scientific Data Summit (NYSDS)

August 15th, 2016

Visualization Research Center
University of Stuttgart

Visualization Research Center
University of Stuttgart

Motivation

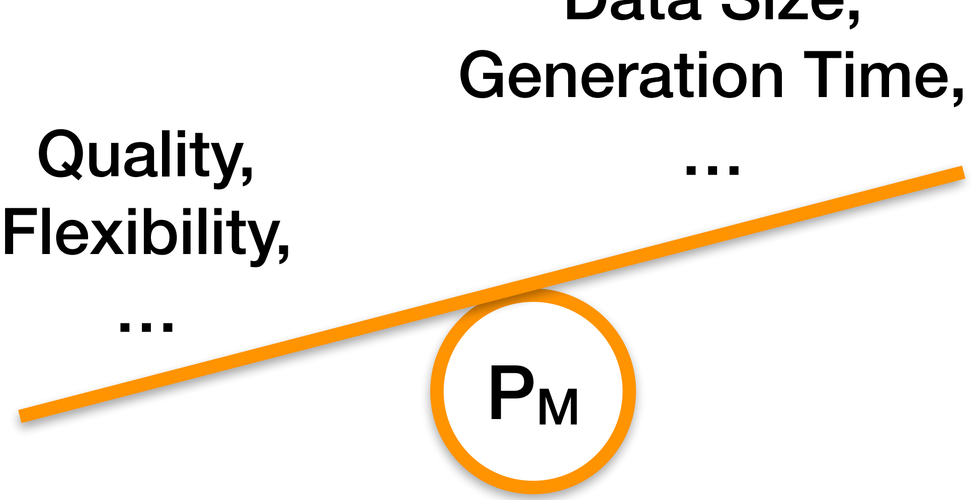
- data sizes increase, e.g.,
  - large-scale supercomputer simulations
  - high resolution acquisition devices

  ➡ compute-data gap
- storing (or moving) all data impractical/impossible
  - consumes a lot of space, time and power
- … but just discarding data also not desirable

- approach: in-situ (on site) reduction of data
  - process data as soon (and where) it is generated
- hybrid in situ visualization
  - create intermediate representations (IR) immediately
  - IR can be used for data exploration later

Motivation

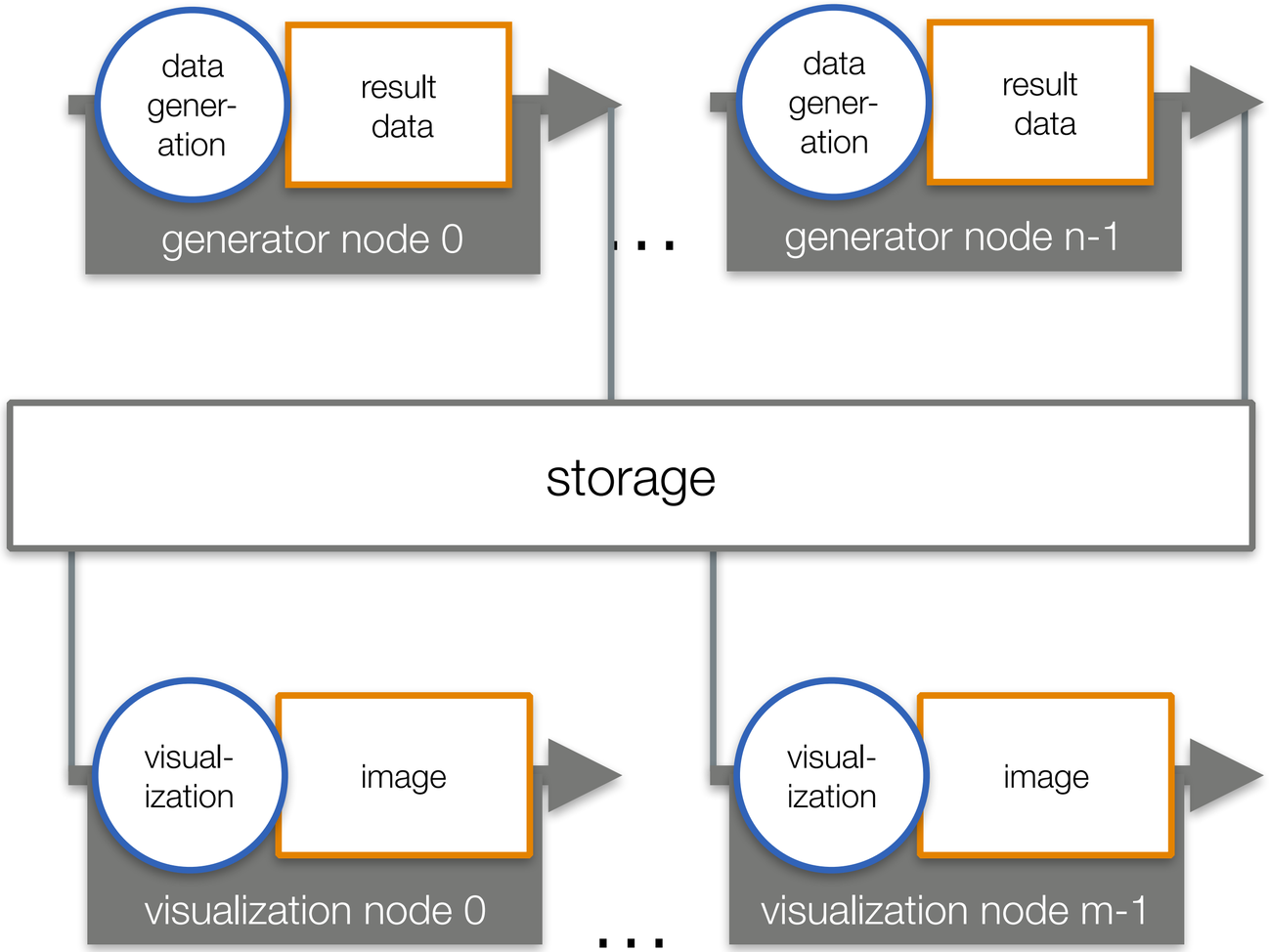
- IR for hybrid in situ visualization should
  - … achieve a significant reduction in size
  - … be producible quickly/efficiently
  - … depict the data of interest
  - … support interactive exploration



- IR generation inherently involves a trade-off
  - typically tunable via parameters $\mathbf{P_M}$
- goal of this work: optimize parameters for IR generation
  - analysis and quantification of their impact
  - auto-tuning for different constraints (like time or data size)
  - at the example of an IR for volume visualization

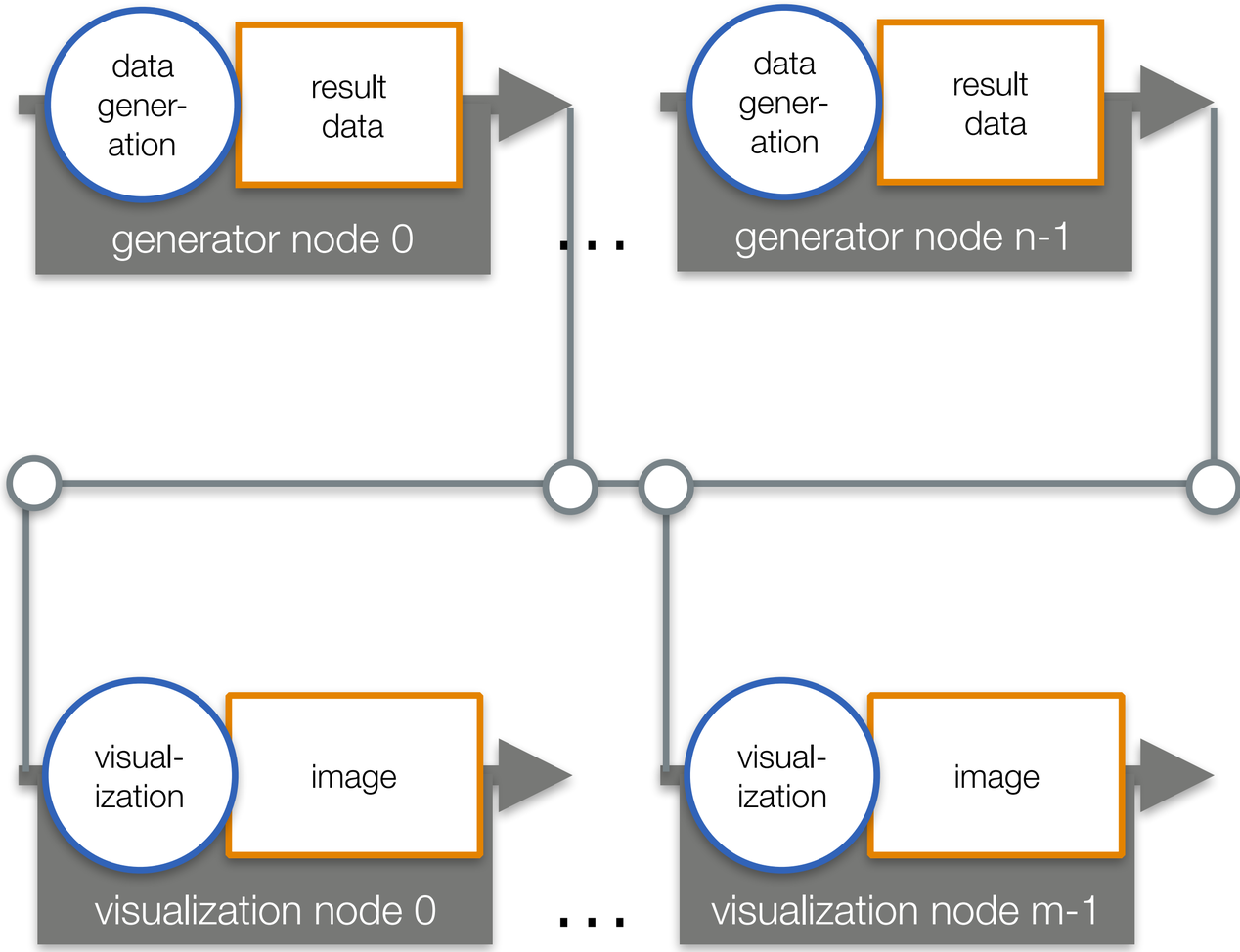
Motivation - In Situ Visualization

- *traditional procedure*
  - decoupled data generation and visualization
  - full (raw) data is stored
    - storage/time/energy constraints limit resolution of stored result data …
    - … but full flexibility preserved for exploring/ analyzing stored data



Motivation - In Situ Visualization

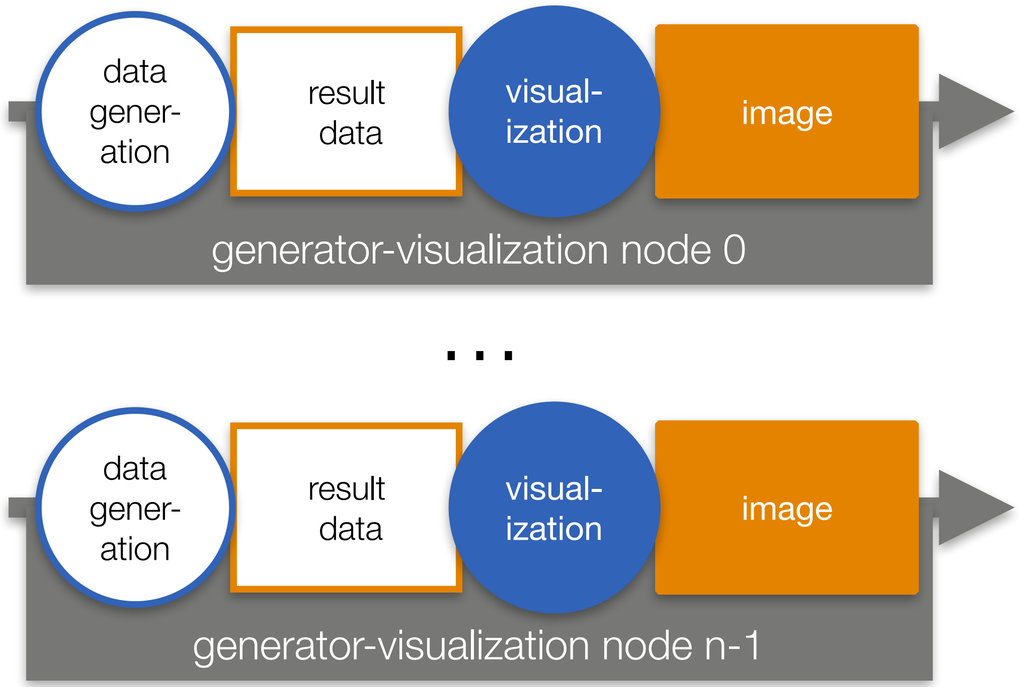
- *in situ visualization w/ loose coupling*
  - concurrent data generation and visualization
    - result data is processed immediately
    - size: visualization results << raw data
    - results become available during simulation run
  - … but on separate resources
    - high network load
    - forced data conversion



loose, tight and hybrid coupling classification according to [Riva et al.] Marzia Rivia, Luigi Caloria, Giuseppa Muscianisia, Vladimir Slavnicb “In-situ Visualization: State-of-the-art and Some Use Cases”, 2014.

Motivation - In Situ Visualization

- *in situ visualization w/ tight coupling*
  - visualization runs on the same cluster node as simulation (in-situ)
  - shared data structures
    - optimally no transfer or data conversion required
  - space decomposition
  - compute resources
  - …
  - the result are images generated by the visualization
    - little flexibility for a posteriori exploration

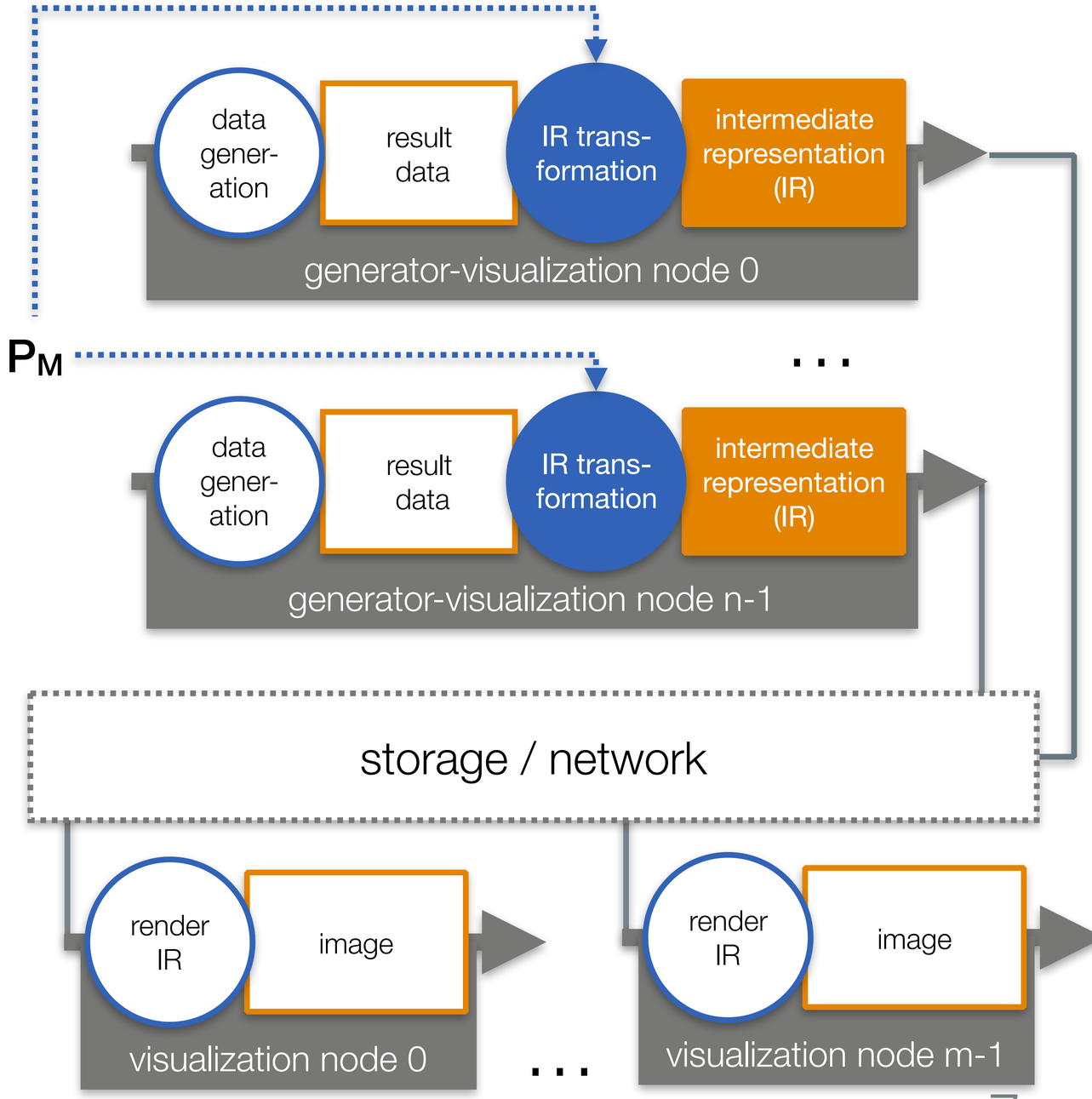


Motivation - In Situ Visualization

- *hybrid in situ visualization (via intermediate representations)*
  - visualization procedure split into two parts

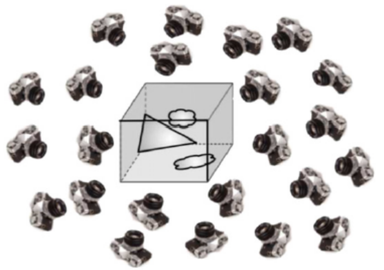
1. transform data into IR
   - tunable via parameters $\mathbf{P_M}$
2. render IR to generate image

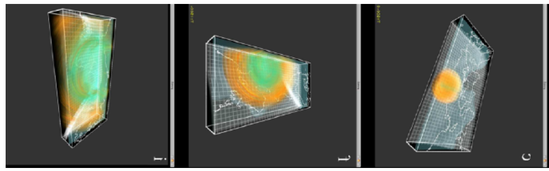
  - combines benefits of traditional method and tight coupling



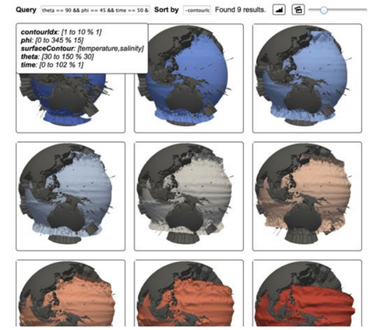
Examples for View-Dependent / Image-based IR for Volume Visualization

- interactive viewing [Kageyama and Yamada]
  - videos recorded different views

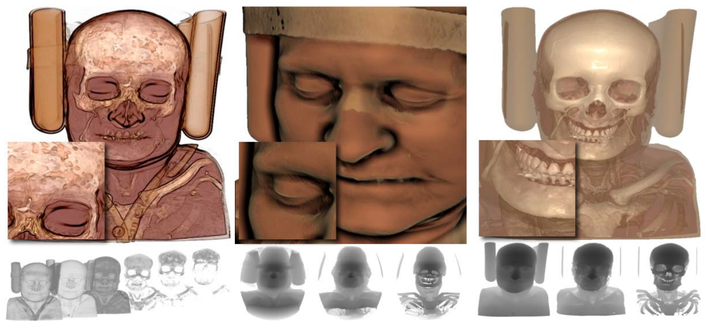




- image data bases [Ahrens et al.]
  - store images created w/ different settings to image database
  - supports content querying and composing new images

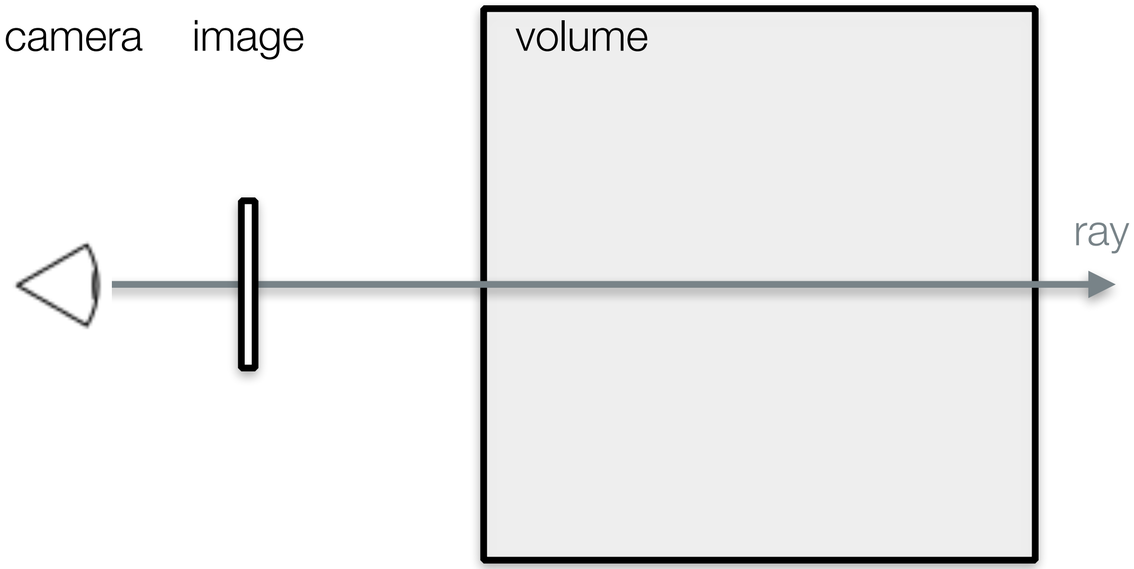


- proxy images [Tikhonova et al.]
  - collection of proxy images containing different information
    - (depth, different view, etc.)
  - proxies are mixed and matched to create renderings



[Tikhonova et al.] Tikhonova, A., Correa, C. D., & Ma, K.-L. (2010). Visualization by Proxy: A Novel Framework for Deferred Interaction with Volume Data. IEEE Transaction on Visualization and Computer Graphics, 16(6), 1551–1559.
[Ahrens et al.] Ahrens, J., Jourdain, S., O'leary, P., Pratchett, J., Rogers, D. H., & Petersen, M. (2014). An Image based Approach to Extreme Scale In Situ Visualization. SC '14 Proceedings, 424–434.
[Kageyama and Yamada] Kageyama, A., & Yamada, T. (n.d.). An approach to exascale visualization: Interactive viewing of in-situ visualization. Computer Physics Communications, 79-85.

Volumetric Depth Images



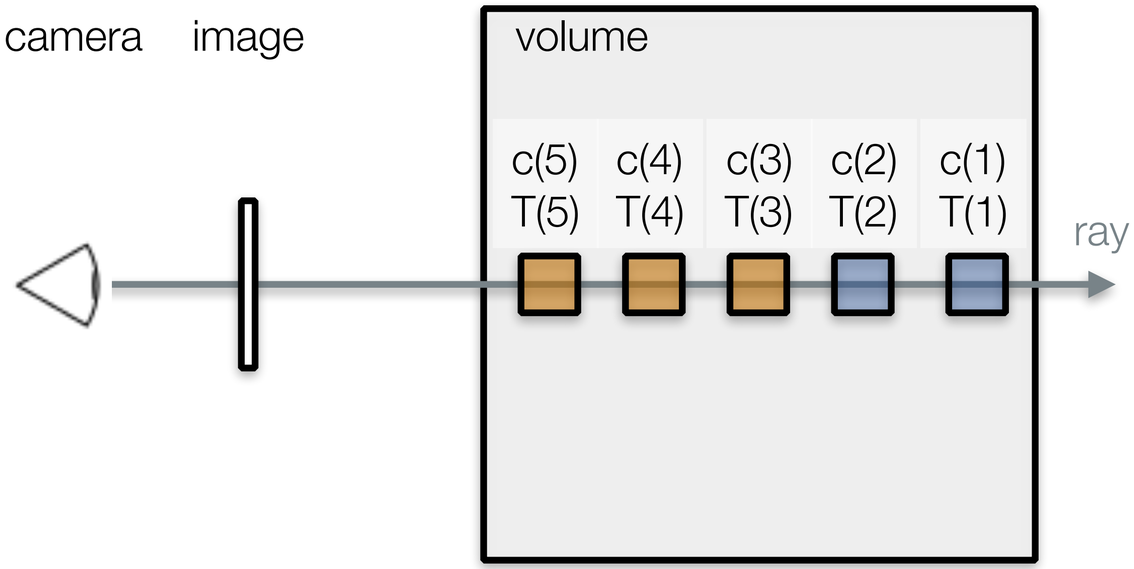
$$J(D) := \sum_{i=1}^{N} c(i) \prod_{j=i+1}^{N} T(j),$$

discretized volume rendering equation

S. Frey, F. Sadlo, and T. Ertl. Explorable volumetric depth images from raycasting, SIBGRAPI, 2013.

Volumetric Depth Images

- volumetric depth images (VDIs) used as IR
  - directly based on volumetric raycasting
    - send one ray through each pixel
    - accumulate color (c) and opacity (T) along ray



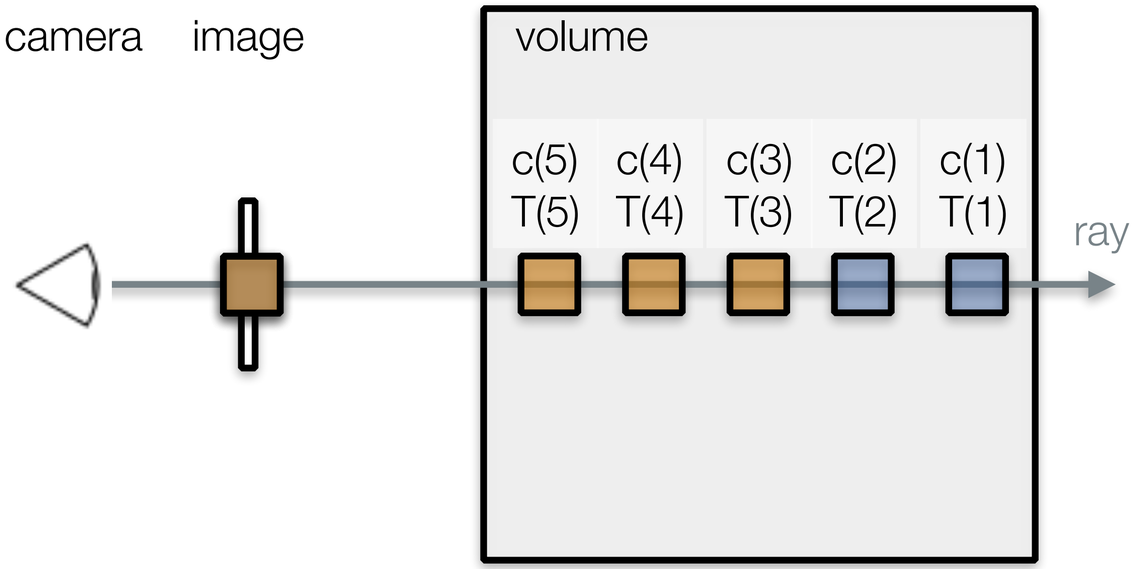
$$J(D) := \sum_{i=1}^{N} c(i) \prod_{j=i+1}^{N} T(j),$$

discretized volume rendering equation

S. Frey, F. Sadlo, and T. Ertl. Explorable volumetric depth images from raycasting, SIBGRAPI, 2013.

Volumetric Depth Images

- volumetric depth images (VDIs) used as IR
  - directly based on volumetric raycasting
    - send one ray through each pixel
    - accumulate color (c) and opacity (T) along ray



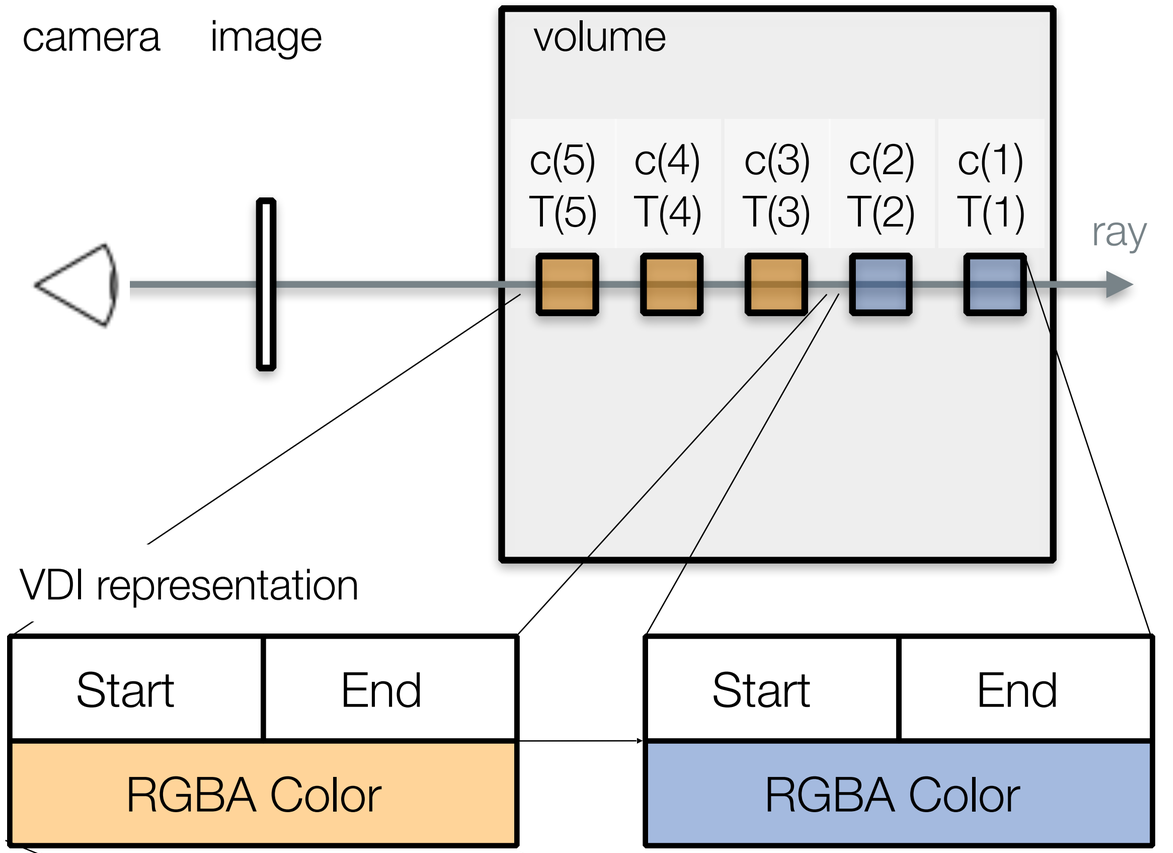
$$J(D) := \sum_{i=1}^{N} c(i) \prod_{j=i+1}^{N} T(j),$$

discretized volume rendering equation

S. Frey, F. Sadlo, and T. Ertl. Explorable volumetric depth images from raycasting, SIBGRAPI, 2013.

Volumetric Depth Images

- volumetric depth images (VDIs) used as IR
  - directly based on volumetric raycasting
    - send one ray through each pixel
    - accumulate color (c) and opacity (T) along ray
- VDIs: partial accumulation of subsequent samples into segments
  - represented by RGBA + 2 depth values
  - can be used for lossy volume reconstruction



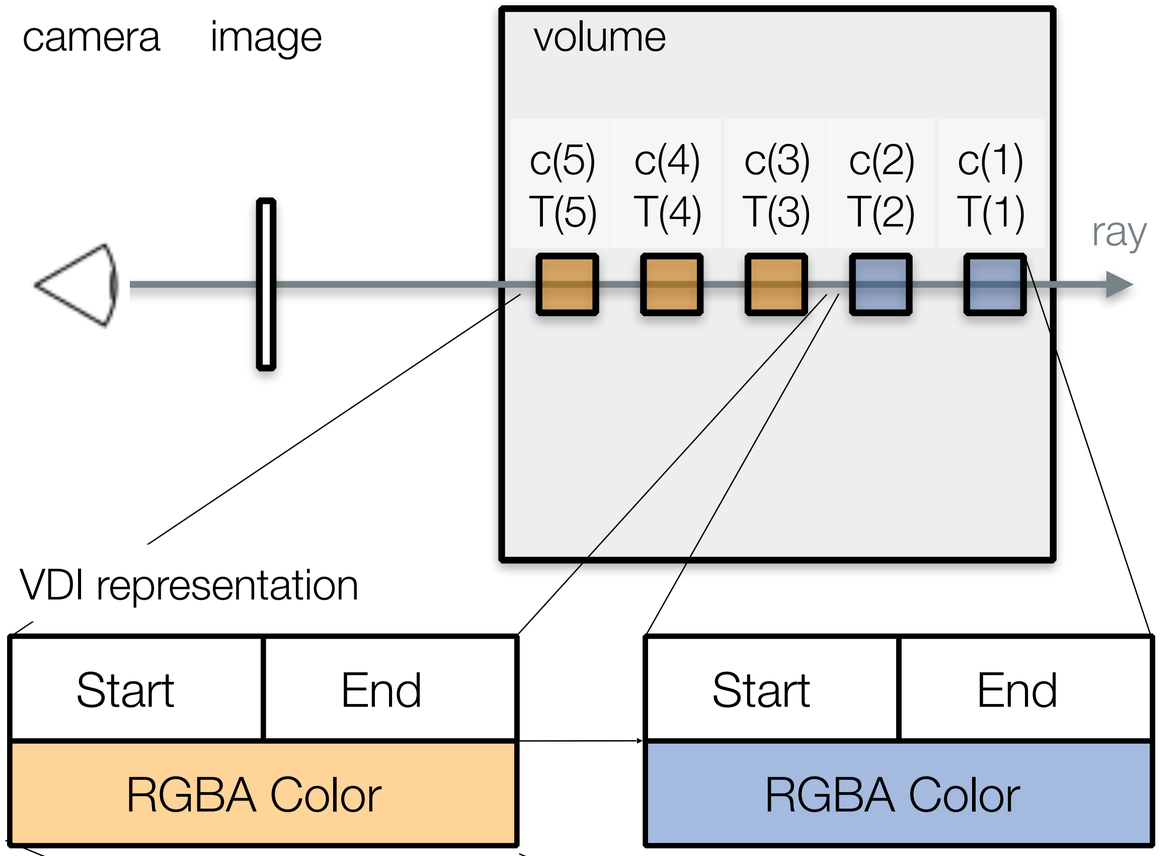
$$J(D) = \sum_{p=1}^{P} \left(\left(\sum_{i=s_b(p)}^{s_f(p)} c(i) \prod_{j=i+1}^{s_f(p)} T(j) \right) \prod_{k=s_f(p)+1}^{N} T(k) \right)$$

discretized volume rendering equation

S. Frey, F. Sadlo, and T. Ertl. Explorable volumetric depth images from raycasting, SIBGRAPI, 2013.

Volumetric Depth Images

- volumetric depth images (VDIs) used as IR
  - directly based on volumetric raycasting
    - send one ray through each pixel
    - accumulate color (c) and opacity (T) along ray
- VDIs: partial accumulation of subsequent samples into segments
  - represented by RGBA + 2 depth values
  - can be used for lossy volume reconstruction
- two parameters in $\mathbf{P_M}$
  - merge threshold **γ**: difference between segment and sample's RGBA color
  - $\mathbf{r}^2$ rays traced (image resolution)



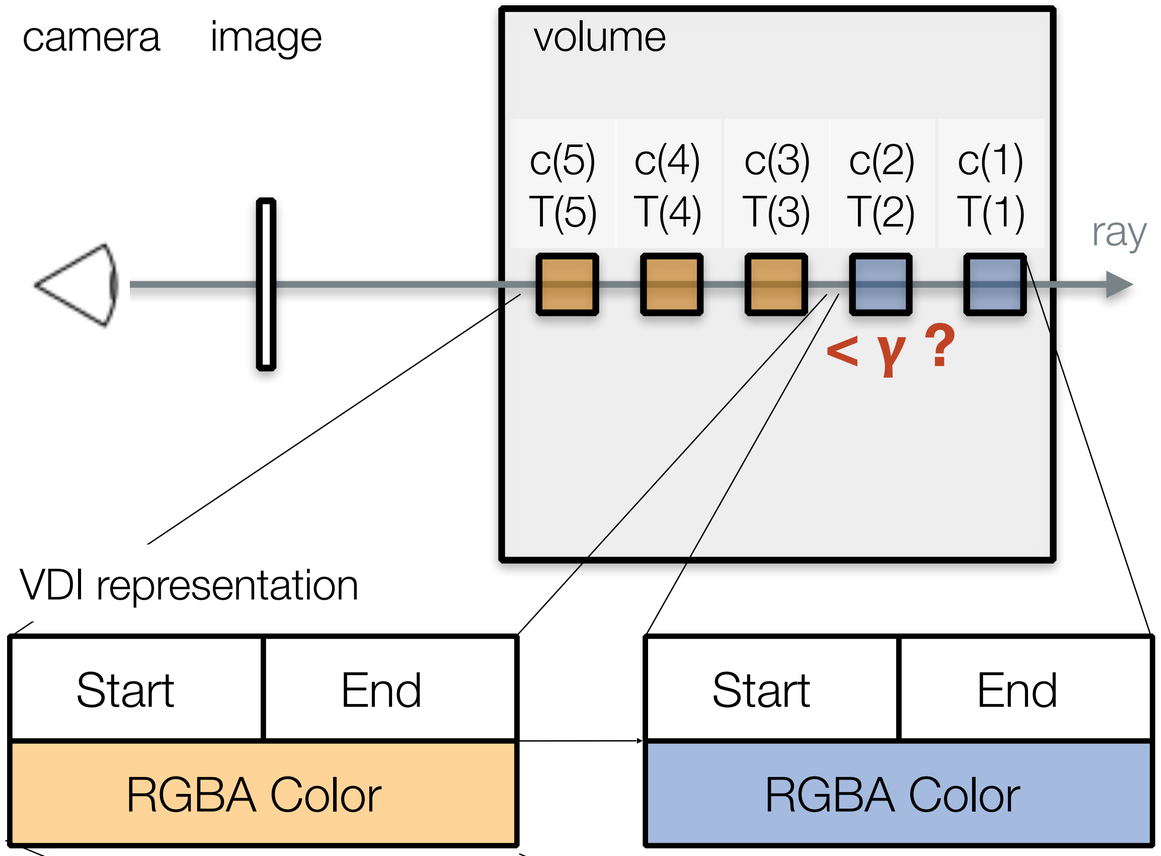
$$J(D) = \sum_{p=1}^{P} \left(\left(\sum_{i=s_b(p)}^{s_f(p)} c(i) \prod_{j=i+1}^{s_f(p)} T(j) \right) \prod_{k=s_f(p)+1}^{N} T(k) \right)$$

discretized volume rendering equation

S. Frey, F. Sadlo, and T. Ertl. Explorable volumetric depth images from raycasting, SIBGRAPI, 2013.

Volumetric Depth Images

- volumetric depth images (VDIs) used as IR
  - directly based on volumetric raycasting
    - send one ray through each pixel
    - accumulate color (c) and opacity (T) along ray
- VDIs: partial accumulation of subsequent samples into segments
  - represented by RGBA + 2 depth values
  - can be used for lossy volume reconstruction
- two parameters in $\mathbf{P_M}$
  - merge threshold **γ**: difference between segment and sample's RGBA color
  - $\mathbf{r}^2$ rays traced (image resolution)



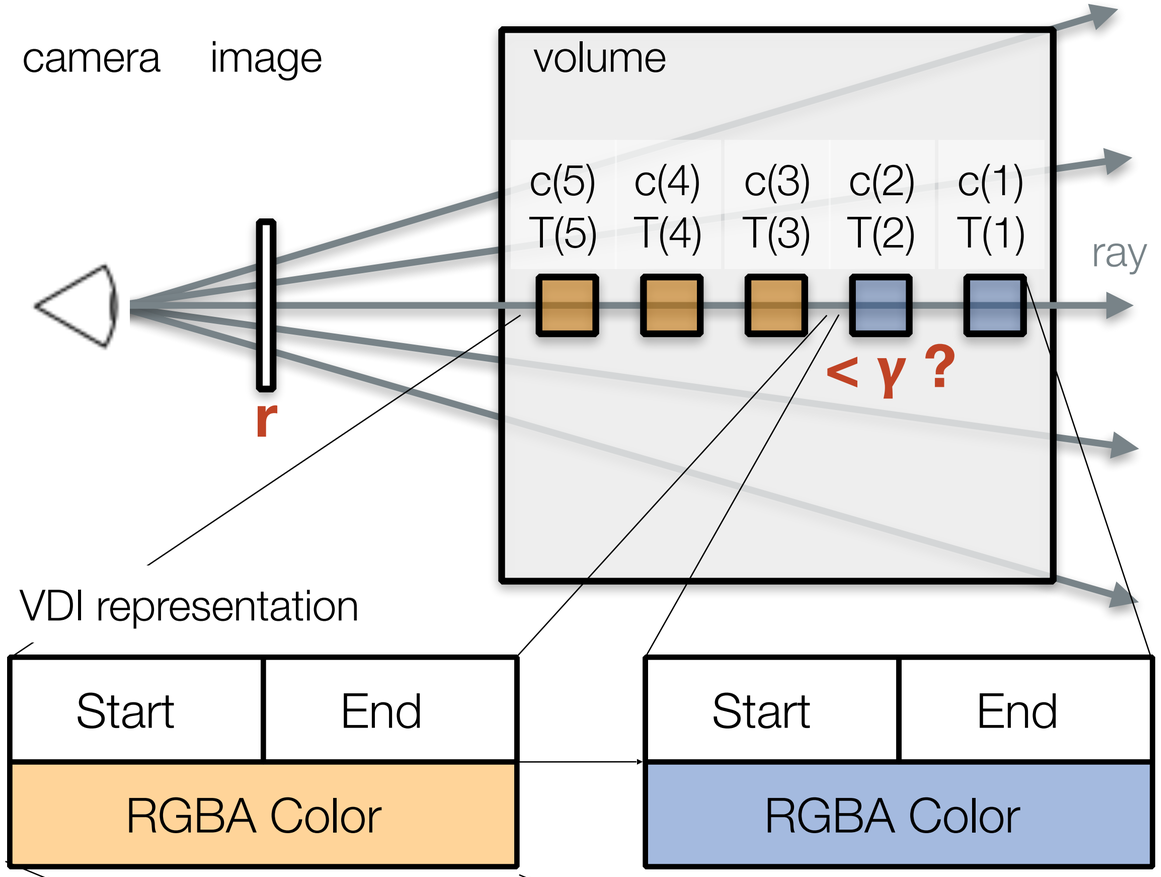
$$J(D) = \sum_{p=1}^{P} \left(\left(\sum_{i=s_b(p)}^{s_f(p)} c(i) \prod_{j=i+1}^{s_f(p)} T(j) \right) \prod_{k=s_f(p)+1}^{N} T(k) \right)$$

discretized volume rendering equation

S. Frey, F. Sadlo, and T. Ertl. Explorable volumetric depth images from raycasting, SIBGRAPI, 2013.

Volumetric Depth Images

- volumetric depth images (VDIs) used as IR
  - directly based on volumetric raycasting
    - send one ray through each pixel
    - accumulate color (c) and opacity (T) along ray
- VDIs: partial accumulation of subsequent samples into segments
  - represented by RGBA + 2 depth values
  - can be used for lossy volume reconstruction
- two parameters in $\mathbf{P_M}$
  - merge threshold **γ**: difference between segment and sample's RGBA color
  - $\mathbf{r}^2$ rays traced (image resolution)



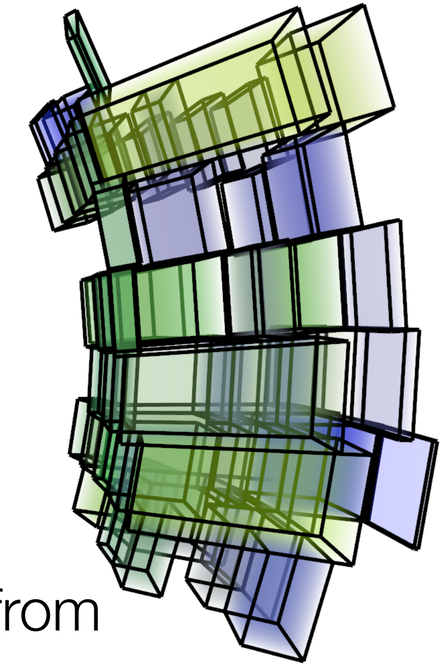
$$J(D) = \sum_{p=1}^{P} \left(\left(\sum_{i=s_b(p)}^{s_f(p)} c(i) \prod_{j=i+1}^{s_f(p)} T(j) \right) \prod_{k=s_f(p)+1}^{N} T(k) \right)$$

discretized volume rendering equation

S. Frey, F. Sadlo, and T. Ertl. Explorable volumetric depth images from raycasting, SIBGRAPI, 2013.

Rendering Volumetric Depth Images

- create frustums from segments
  - geometric representation
  - using RGBA+depth information
  - camera configuration of original view



- rendered using OpenGL & GLSL
  - alpha-blending with color and opacity from the segments
    - contribution of each frustum adjusted w.r.t. length of view ray passing through it

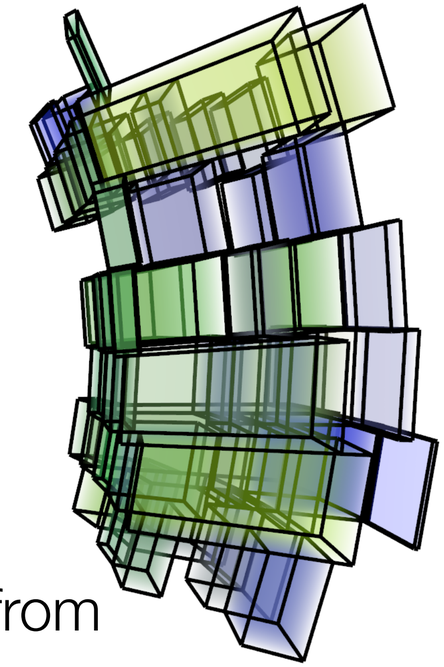
- perfect results for original camera configuration
  - deviations increase with view angle

raycasting 80M cells in 275ms per frame

VDI rendering of 1M frustums in 34ms per frame

Rendering Volumetric Depth Images

- create frustums from segments
  - geometric representation
  - using RGBA+depth information
  - camera configuration of original view



- rendered using OpenGL & GLSL
  - alpha-blending with color and opacity from the segments
    - contribution of each frustum adjusted w.r.t. length of view ray passing through it

- perfect results for original camera configuration
  - deviations increase with view angle

raycasting 80M cells in 275ms per frame

VDI rendering of 1M frustums in 34ms per frame

Auto-Tuning IR

- automatically find best parameters $\mathbf{P_M}$ for IR
  - at example of volume visualization and VDIs
- overview: four phases
  - setup
    - specify analysis scenario as input for tuning
  - generate new parameter candidates
    - tuning loop until no further parameter values
  - obtain performance indicators
    - quantify performance in different regards
  - identify best parameter setting
    - condense performance value to single indicator

 # setup
- scene and render parameters $\mathbf{P_R}$

 # generate new parameter candidates
while (new params $\mathbf{P_M}$) ***loop***

> **# obtain performance indicators**
> - $\boldsymbol{\tau}$ ← generate intermediate representation **M** ($\mathbf{P_M}$)
> - $\boldsymbol{\sigma}$ ← | **M** |
> - **q** ← assess rendering quality (**M**, $\mathbf{P_R}$)
>
> **# identify best parameter setting**
> - **α** ← utility evaluation **ν** ($\boldsymbol{\sigma}$, **q**, $\boldsymbol{\tau}$)
> - **if** α < α'
>
> α' ← α
>
> $\mathbf{P'_M}$ ← $\mathbf{P_M}$

end loop

Auto-Tuning IR

- setup
  - data set & transfer function
  - camera position and orientation

- generate new test parameters $\mathbf{P_M}$
  - parameter ranges from prior experiments
  - for VDIs $\mathbf{P_M} = (\boldsymbol{\gamma}, \mathbf{r})$:
    - interrupt parameter $\boldsymbol{\gamma}$
      - similarity of combined samples in ray space
    - image resolution $\mathbf{r}$
      - number of rays traced (image space)

 # setup

- scene and render parameters $\mathbf{P_R}$

 # generate new parameter candidates

while (new params $\mathbf{P_M}$) ***loop***

> **# obtain performance indicators**
> - $\tau \leftarrow$ generate intermediate representation **M** ($\mathbf{P_M}$)
> - $\sigma \leftarrow |\ \mathbf{M}\ |$
> - **q** $\leftarrow$ assess rendering quality (**M, P**$_R$)
>
> **# identify best parameter setting**
> - $\alpha \leftarrow$ utility evaluation **v** (σ, **q,** τ)
> - **if** $\alpha < \alpha'$
>
> $\alpha' \leftarrow \alpha$
>
> $P'_M \leftarrow P_M$

end loop

Auto-Tuning IR

- obtain IR and performance indicators (for $\mathbf{P_M}$)
  - three quantities: size $\boldsymbol{\sigma}$, time $\boldsymbol{\tau}$, and quality **q**
  - time $\boldsymbol{\tau}$ to generate intermediate representation **M**
    - amount of time (VDI) generation takes
  - storage cost $\boldsymbol{\sigma}$
    - size of representation compressed via bzip2
  - rendering quality **q**
    - create images with **M** …
    - …and reference renderings (via volume raycasting)
    - assess using image quality metric (PSNR)
    - here: for rotation angles 10° - 60° (step size 10°)

 # setup
- scene and render parameters $\mathbf{P_R}$

 # generate new parameter candidates
while (new params $\mathbf{P_M}$) ***loop***

> **# obtain performance indicators**
> - $\boldsymbol{\tau}$ ← generate intermediate representation **M** ($\mathbf{P_M}$)
> - $\boldsymbol{\sigma}$ ← | **M** |
> - **q** ← assess rendering quality (**M**, $\mathbf{P_R}$)
>
> **# identify best parameter setting**
> - α ← utility evaluation **v** ($\boldsymbol{\sigma}$, **q**, $\boldsymbol{\tau}$)
> - **if** $\alpha < \alpha'$
>
> $\alpha' \leftarrow \alpha$
> $P'_M \leftarrow P_M$

end loop

Auto-Tuning Intermediate Representations (VDIs)

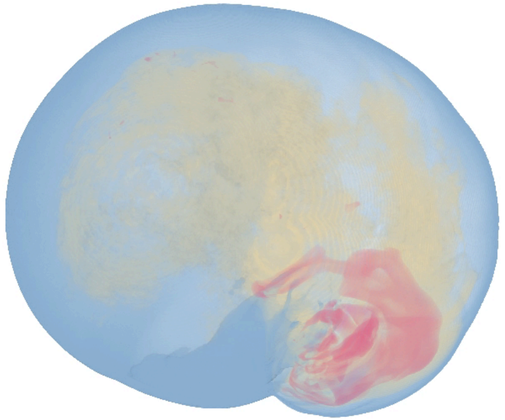
- identify best parameter setting
  - utility function **ν** gives single performance value **α**
    - based on data size ***σ***,
    - rendering quality **q**,
    - and IR generation time ***τ***
  - we optimize **q** for target limited w.r.t.
    - data size ***σ*** (e.g., 2 MB per IR)
    - time ***τ*** (e.g., 1 second per IR)
- keep $\mathbf{P_M}$ that yields the smallest **α**

```
# setup
▪ scene and render parameters P_R

# generate new parameter candidates
while (new params P_M) loop

    # obtain performance indicators
    ▪ τ ← generate intermediate representation M (P_M)
    ▪ σ ← | M |
    ▪ q ← assess rendering quality (M, P_R)

    # identify best parameter setting
    ▪ α ← utility evaluation ν (σ, q, τ)
    ▪ if α < α'
        α' ← α
        P'_M ← P_M

end loop
```

Results

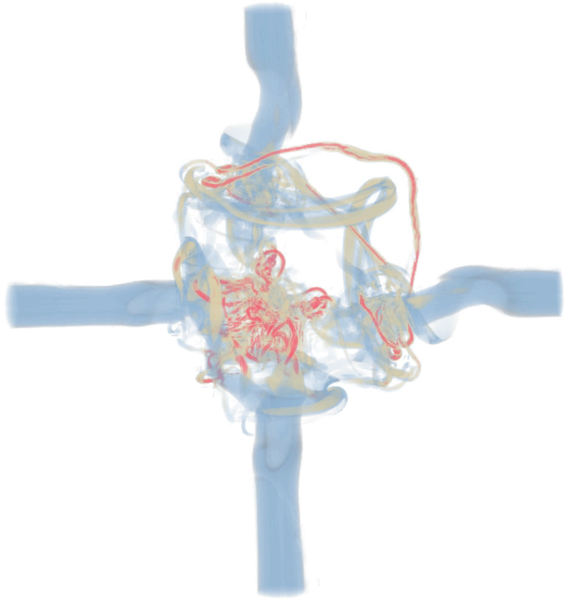
- three data sets
  - supernova (two time steps from simulation, 432^3)
  - λ_2 (vortex extraction from CFD simulation, 529^3)
  - zeiss (CT scan, 680^3)

- VDIs are used as IR, processed on GPU
  - generated via modified CUDA-based raycaster
  - rendered via OpenGL / GLSL

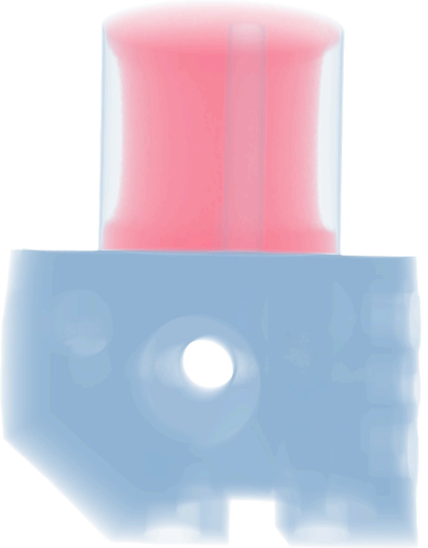


supernova (t=40)
(source: Kwan-Liu Ma, UC Davis)

supernova (t=20)



λ_2



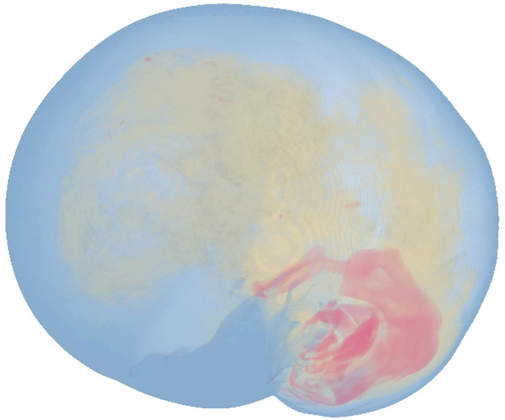
zeiss
(source: Daimler AG)

reference image resolution: 768^2
tuning machine hardware : NVIDIA GTX980 / Intel Core i7-3820 / 16GB RAM

Results

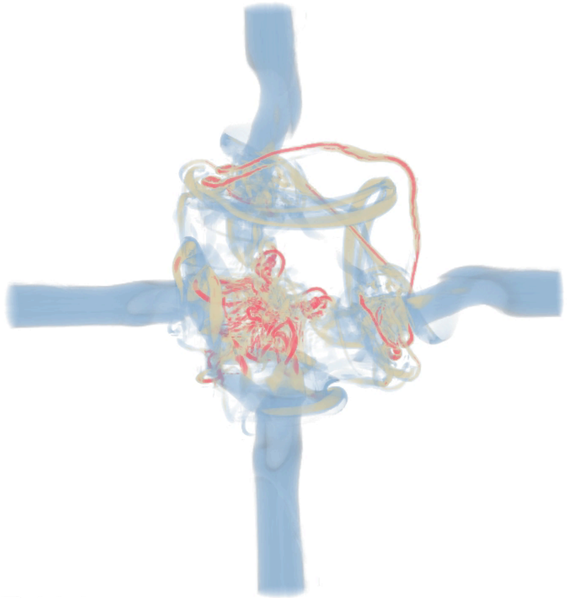
- three data sets
  - supernova (two time steps from simulation, 432^3)
  - λ_2 (vortex extraction from CFD simulation, 529^3)
  - zeiss (CT scan, 680^3)

- VDIs are used as IR, processed on GPU
  - generated via modified CUDA-based raycaster
  - rendered via OpenGL / GLSL

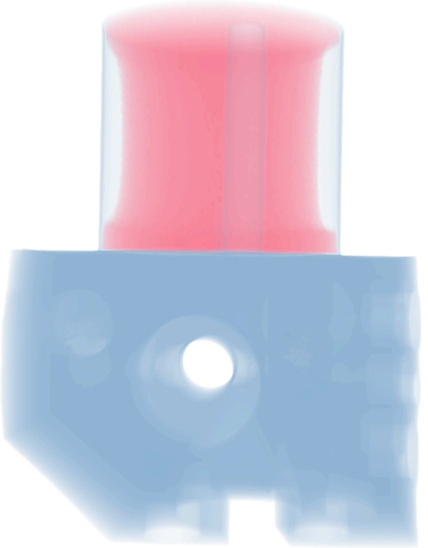


supernova (t=40)
(source: Kwan-Liu Ma, UC Davis)

supernova (t=20)



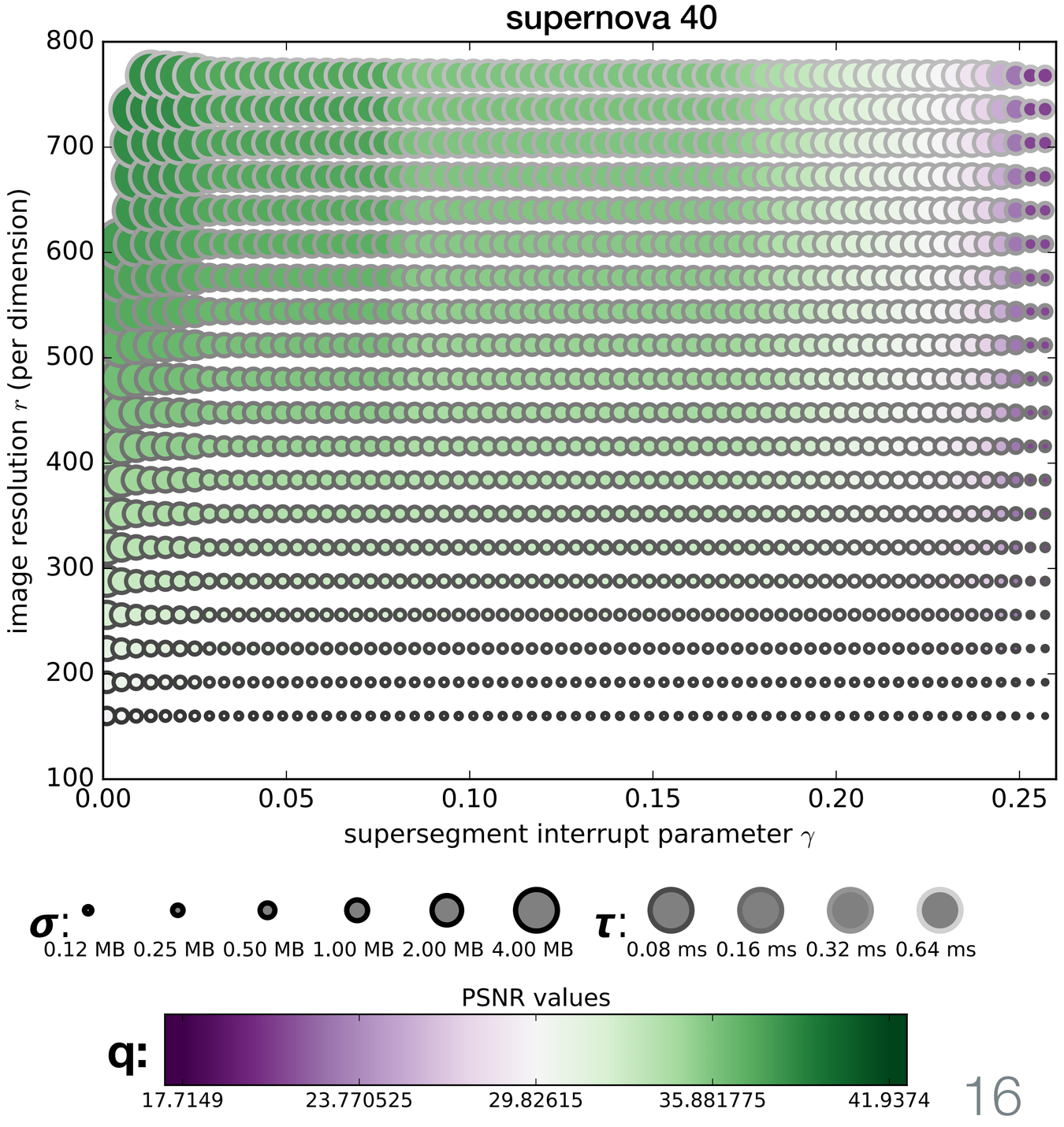
λ_2



zeiss
(source: Daimler AG)

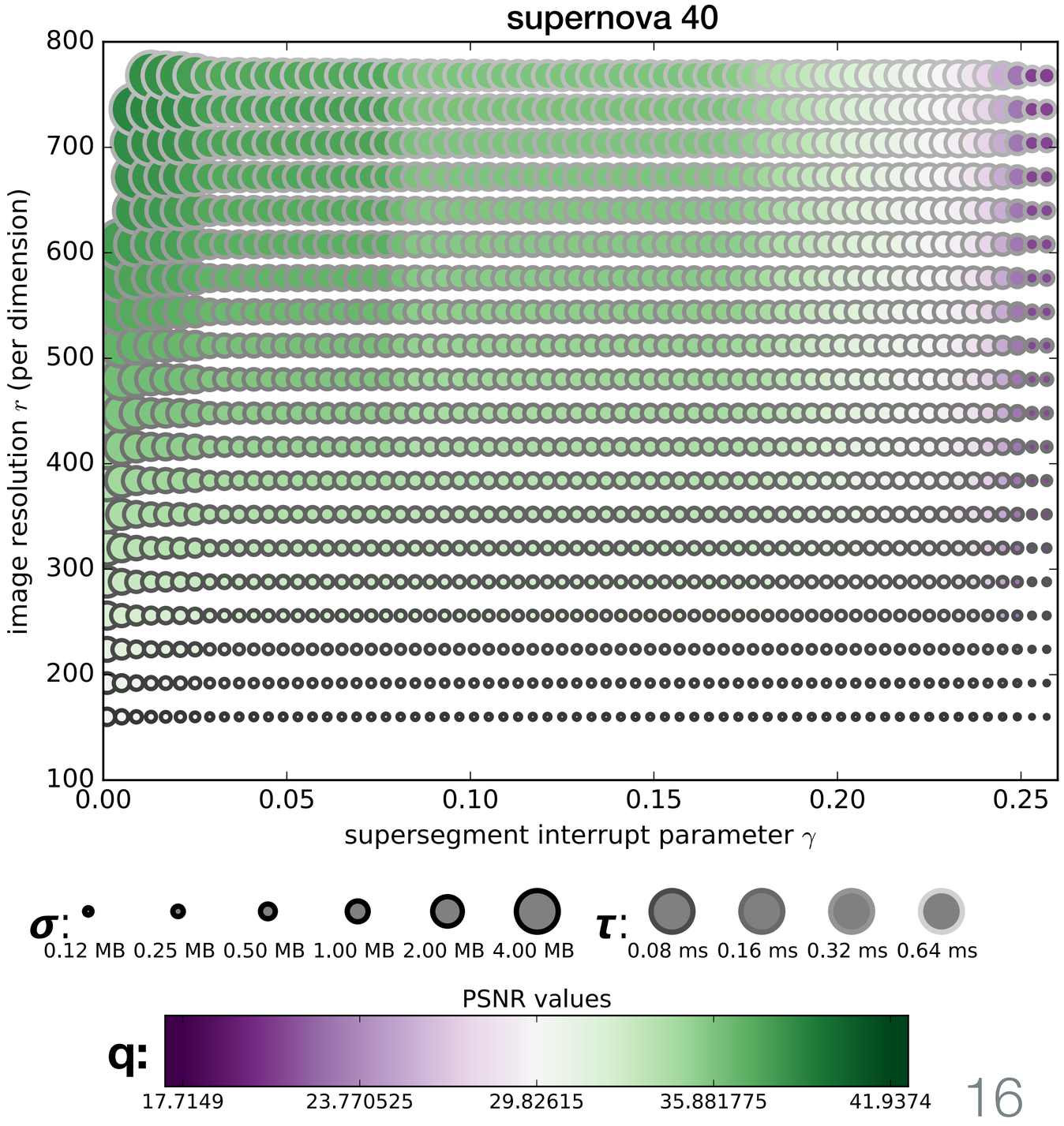
reference image resolution: 768^2
tuning machine hardware : NVIDIA GTX980 / Intel Core i7-3820 / 16GB RAM

Parameter Study - Supernova 40



Parameter Study - Supernova 40

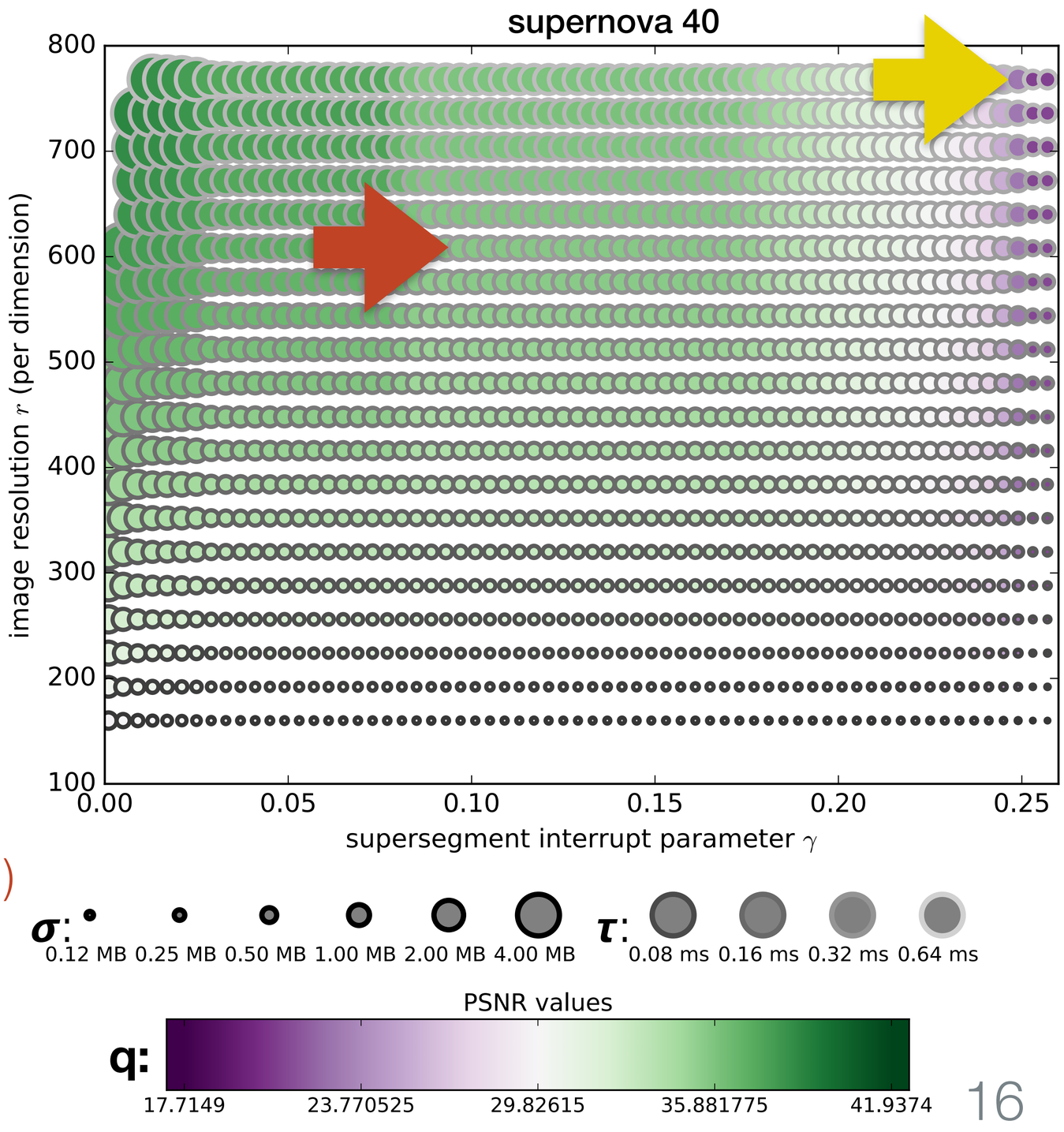
- parameter ranges for $\mathbf{P_M} = (\mathbf{r}, \boldsymbol{\gamma})$
  - **r**: 160-768, step size 32
  - **γ**: 0.001-0.26, step size 0.004
- we consider size $\boldsymbol{\sigma}$, time $\boldsymbol{\tau}$, and quality **q**
  - larger **r** / smaller **γ** → higher **q** and $\boldsymbol{\sigma}$
  - $\boldsymbol{\tau}$ mostly influenced by **r**
    - scales approximately linear with #rays



Parameter Study - Supernova 40

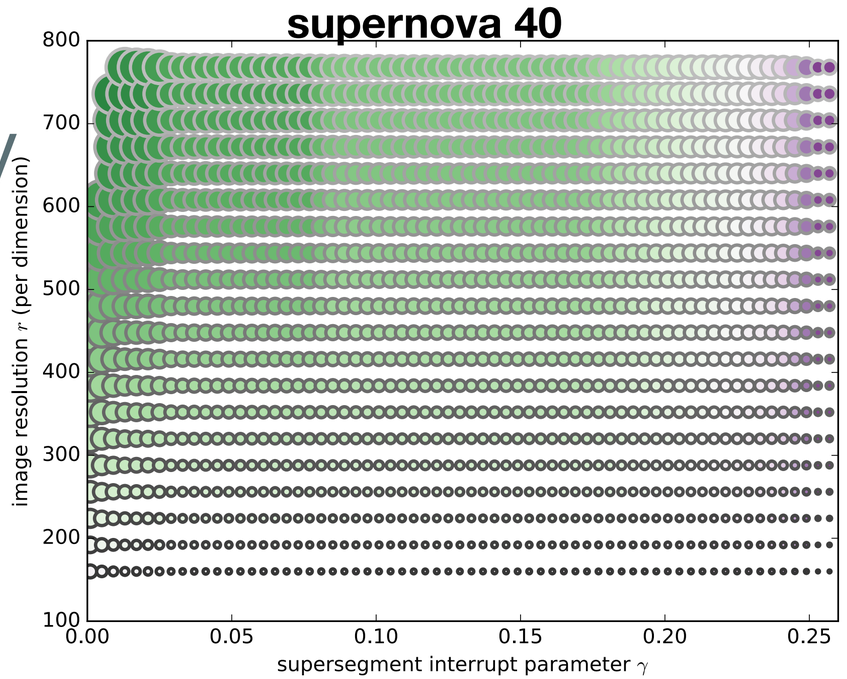
- parameter ranges for $\mathbf{P_M} = (\mathbf{r}, \boldsymbol{\gamma})$
  - **r**: 160-768, step size 32
  - **γ**: 0.001-0.26, step size 0.004
- we consider size $\boldsymbol{\sigma}$, time $\boldsymbol{\tau}$, and quality **q**
  - larger **r** / smaller **γ** → higher **q** and $\boldsymbol{\sigma}$
  - $\boldsymbol{\tau}$ mostly influenced by **r**
    - scales approximately linear with #rays

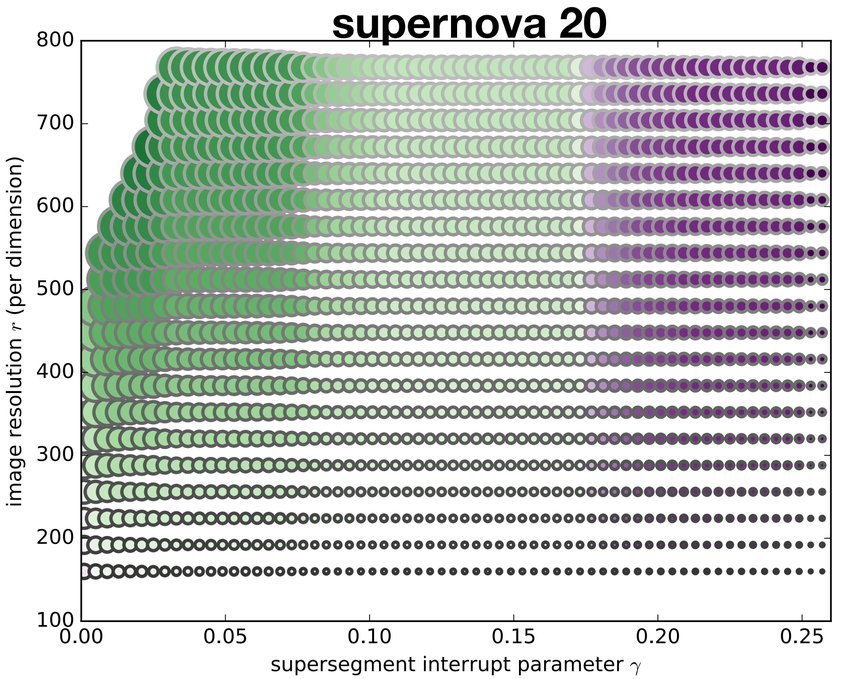
- some parameter configurations better
  - e.g., (**r**=768, **γ**=0.25) vs. (**r**=608, **γ**=0.11)
  - approximately same data size …
  - … but higher quality for (**r**=608, **γ**=0.11)

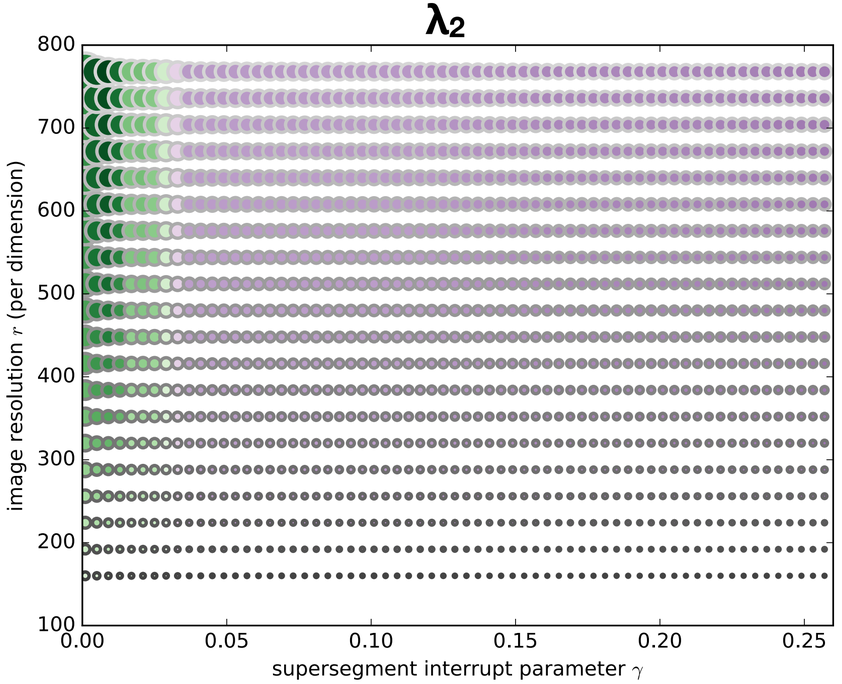


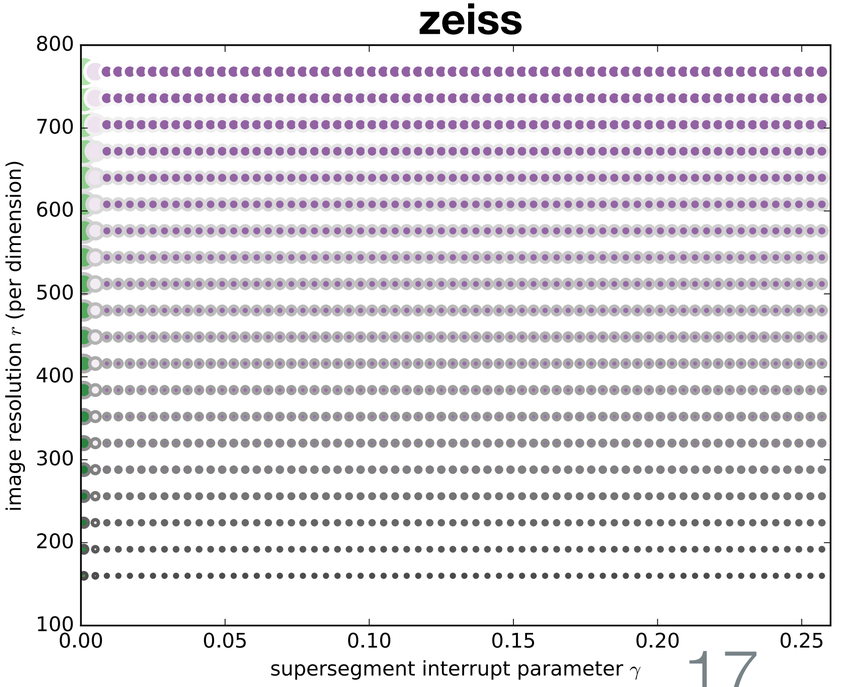
Parameter Study

- characteristics of VDIs are data-dependent
  - supernova, λ_2, and zeiss differ significantly
  - supernovas are similar
    - 40 with higher complexity
- relative impact of **r** is fairly similar …
- … but **γ** differs
  - leaps for “boundaries” in data









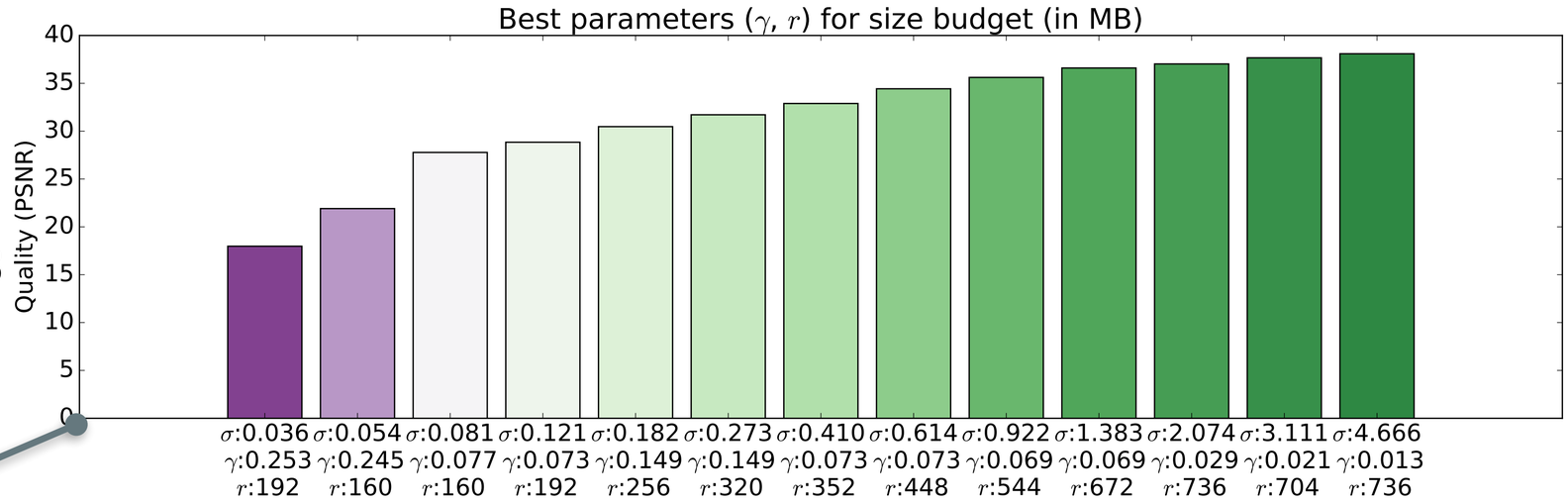
Auto-Tuning

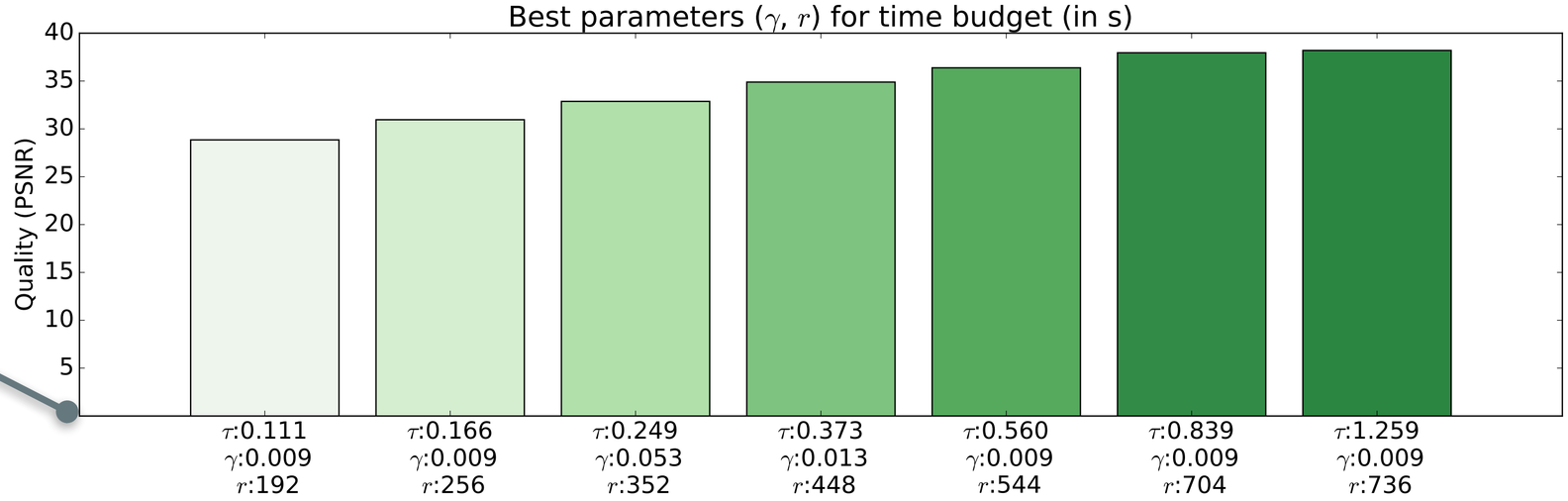
- reach best possible quality for
  - limited size ***σ***
    - between 0.036 MB and 4.666 MB
  - limited IR generation time ***τ***
    - between 0.111 s and 1.259 s

- increasing size limit…
  - … higher quality
  - … both **r** and **γ** increase

- increasing time budget …
  - … higher quality
  - … particularly **r** is increased
  - … **γ** is relatively small

supernova 40

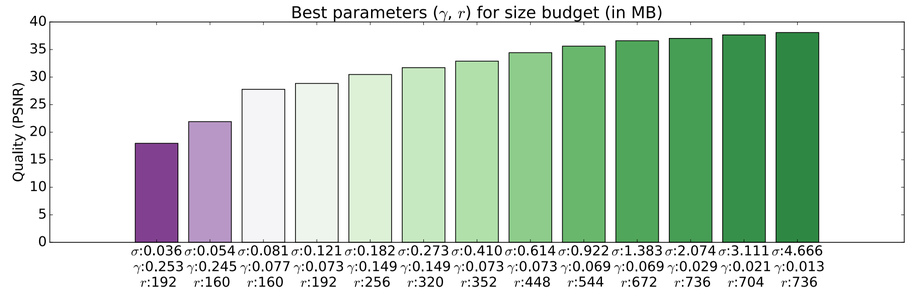


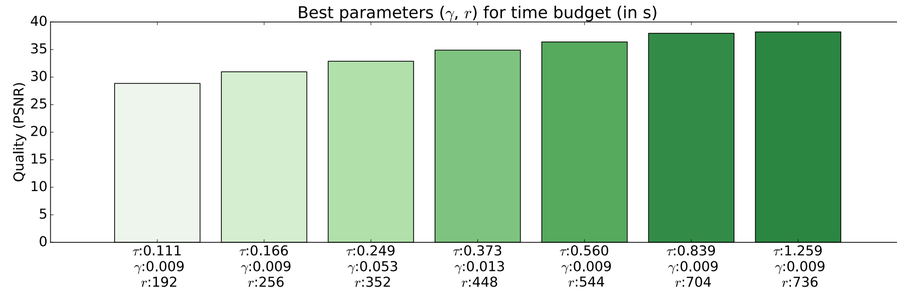


Auto-Tuning

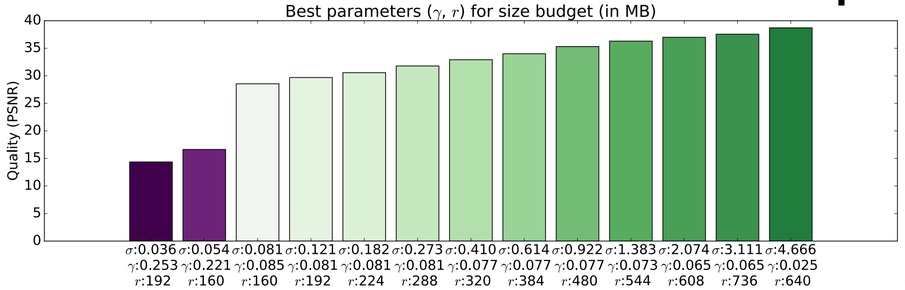
- generally increasing quality with increasing budget …
- … yet individual behavior heavily depends on data

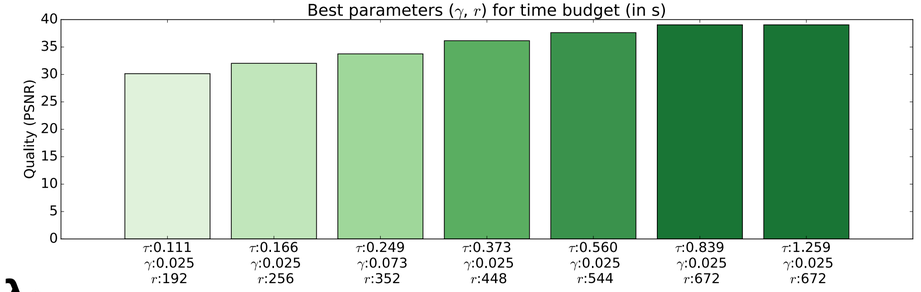
size **supernova 40** *time*



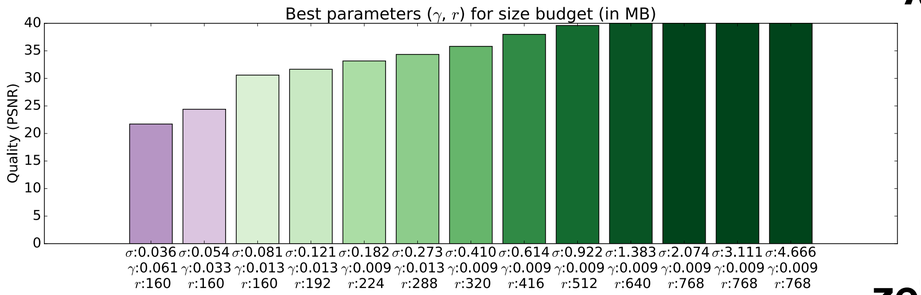


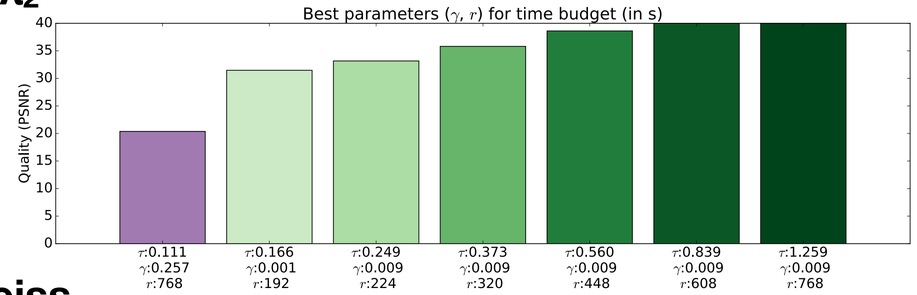
supernova 20



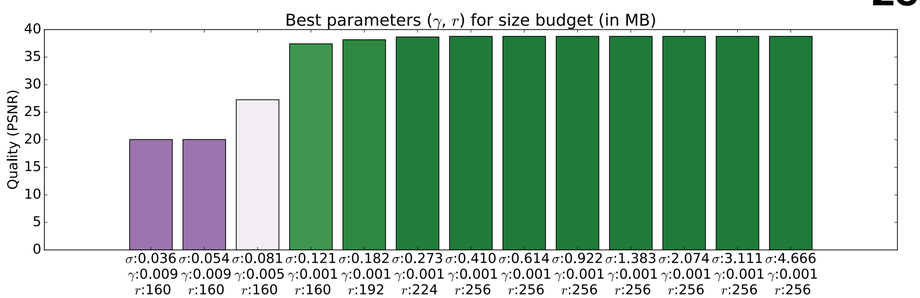


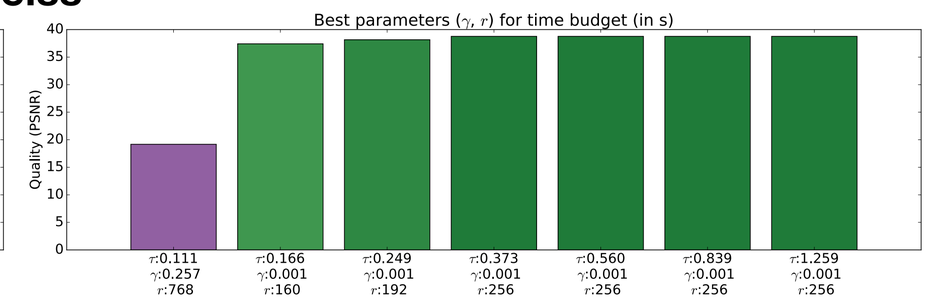
λ_2



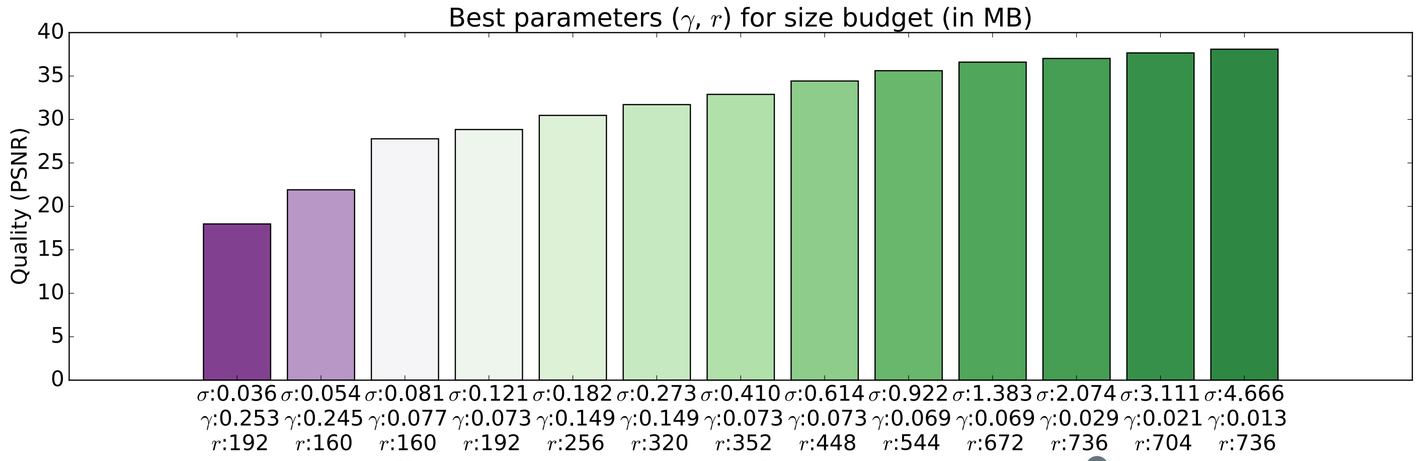


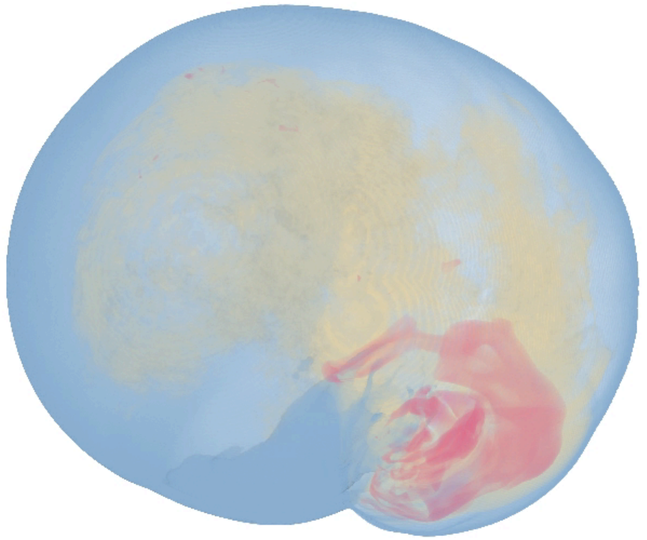
zeiss



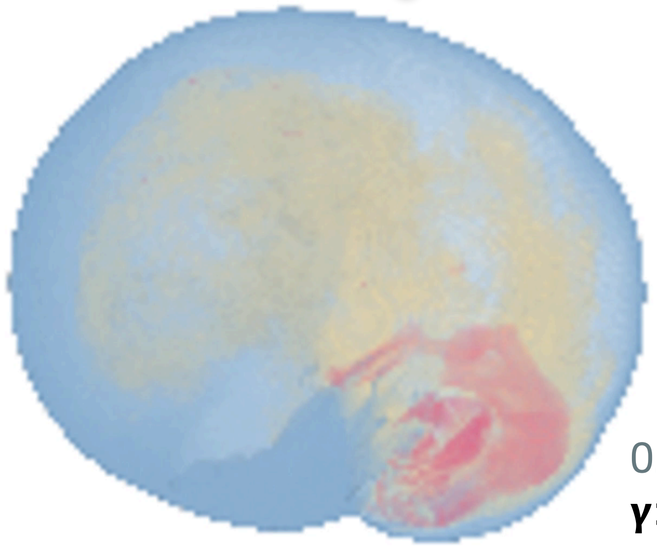


Auto-Tuning - Supernova 40

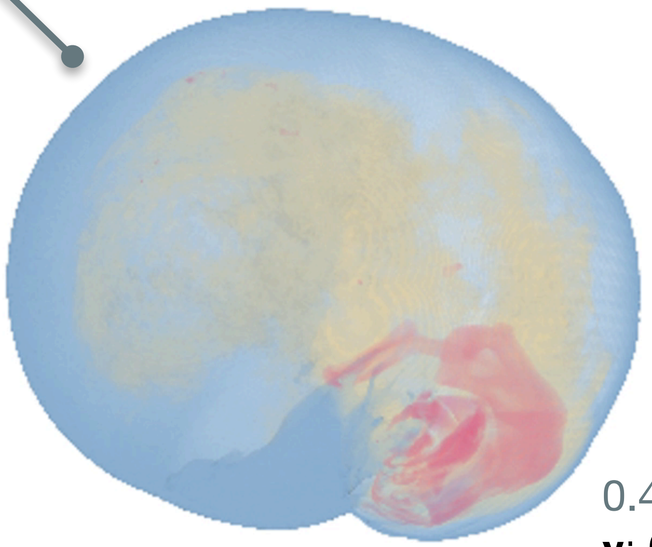




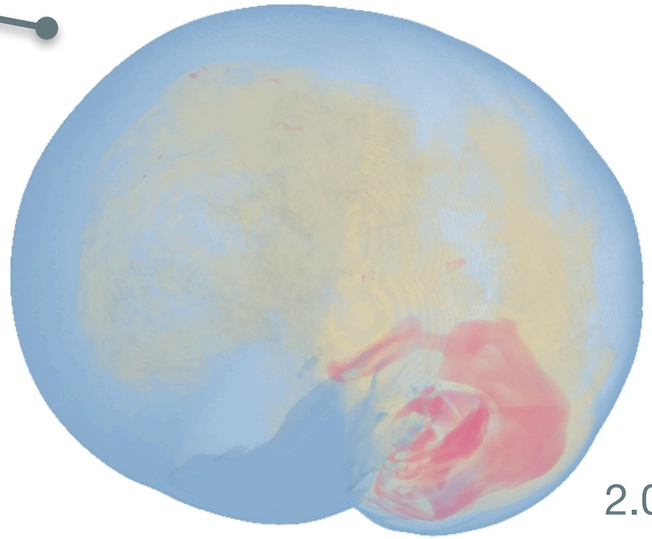
ref



0.081 MB
γ: 0.077
r: 160

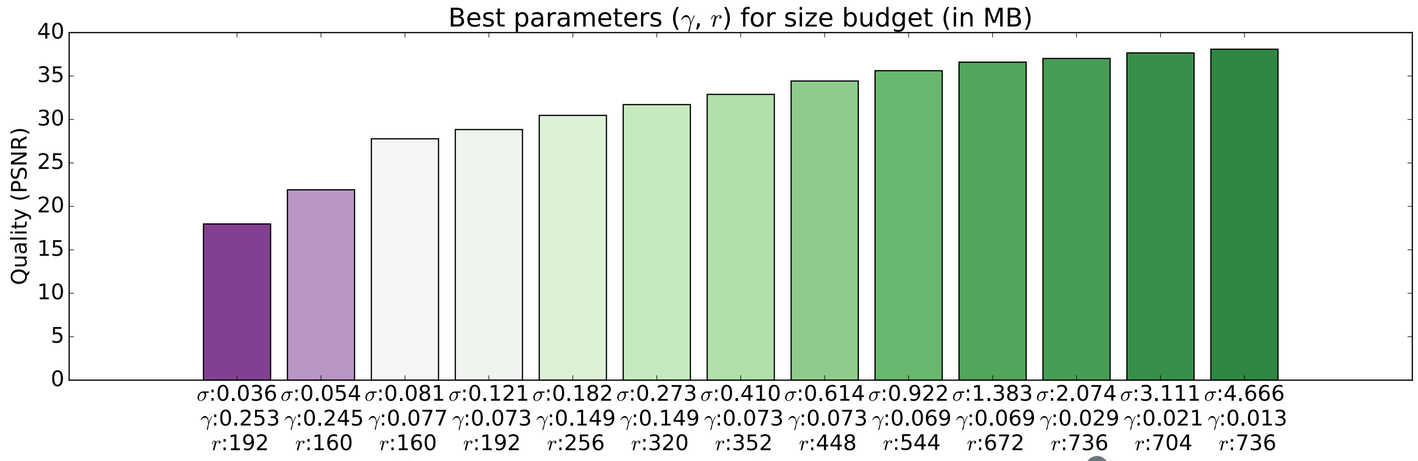


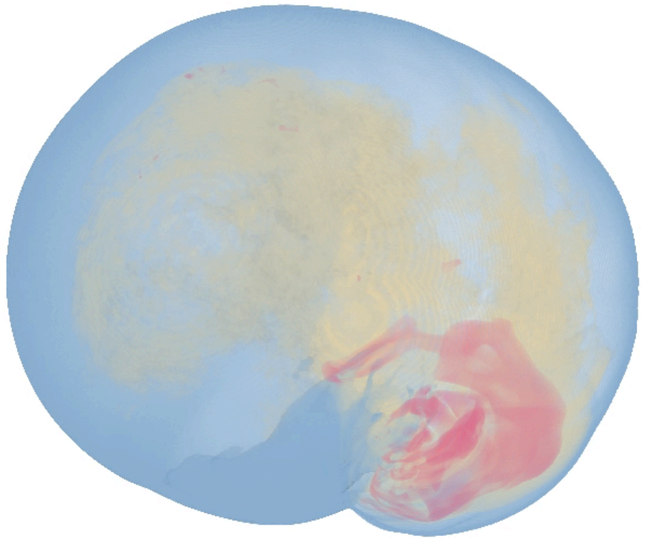
0.41 MB
γ: 0.073
r: 352



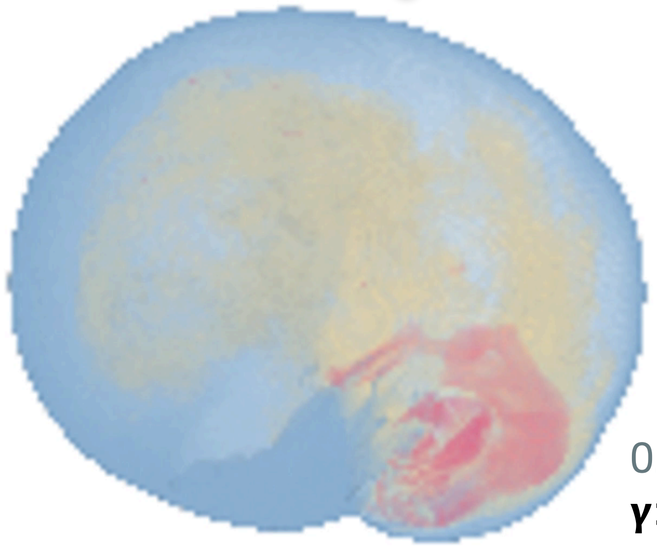
2.074 MB
γ: 0.029
r: 736

Auto-Tuning - Supernova 40

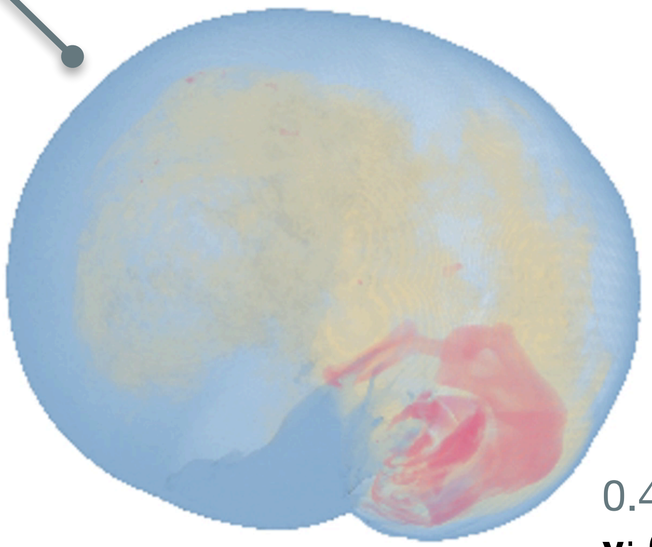




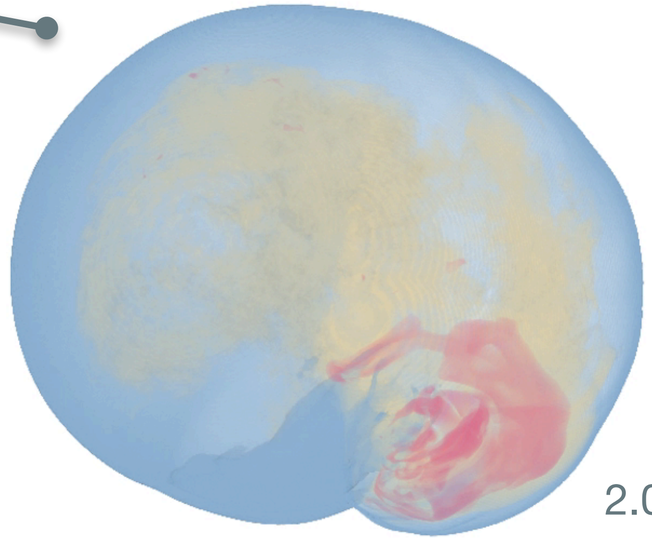
ref



0.081 MB
γ: 0.077
r: 160

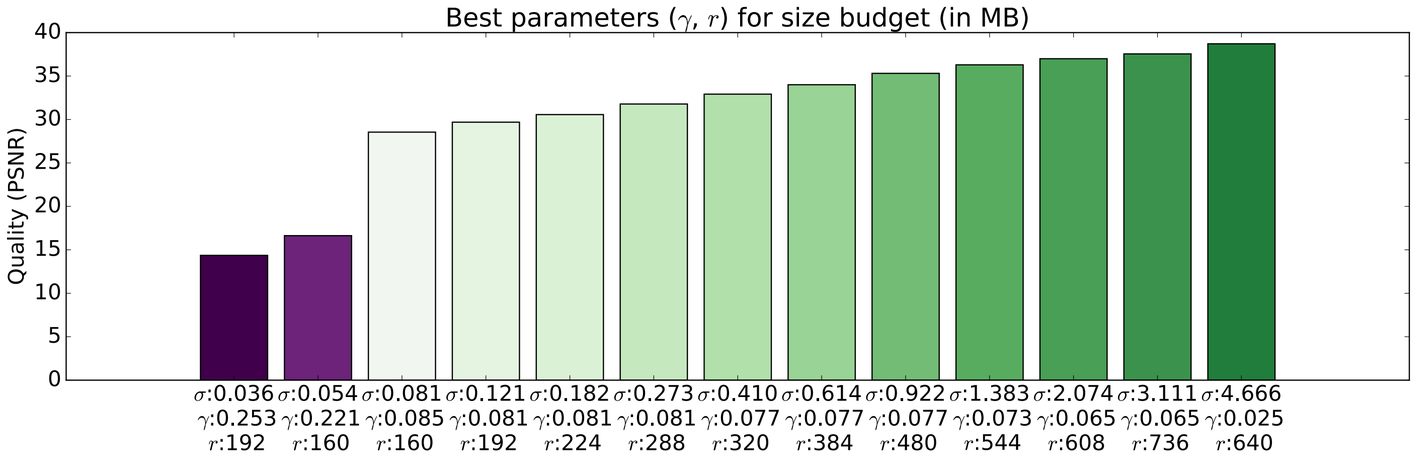


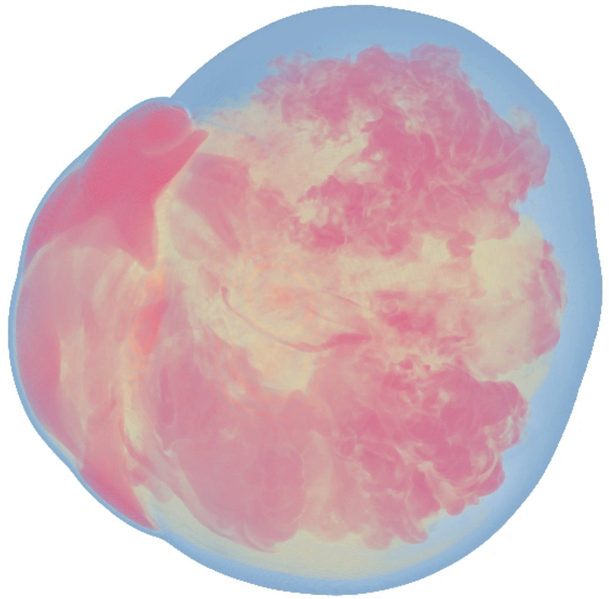
0.41 MB
γ: 0.073
r: 352



2.074 MB
γ: 0.029
r: 736

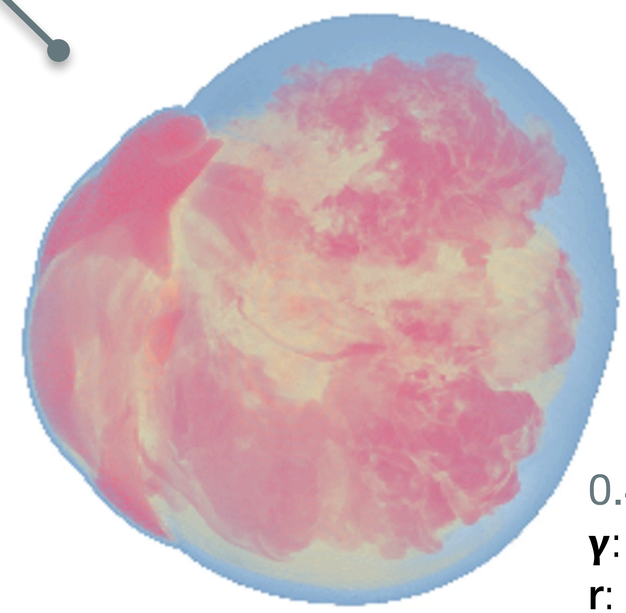
Auto-Tuning - Supernova 20





ref

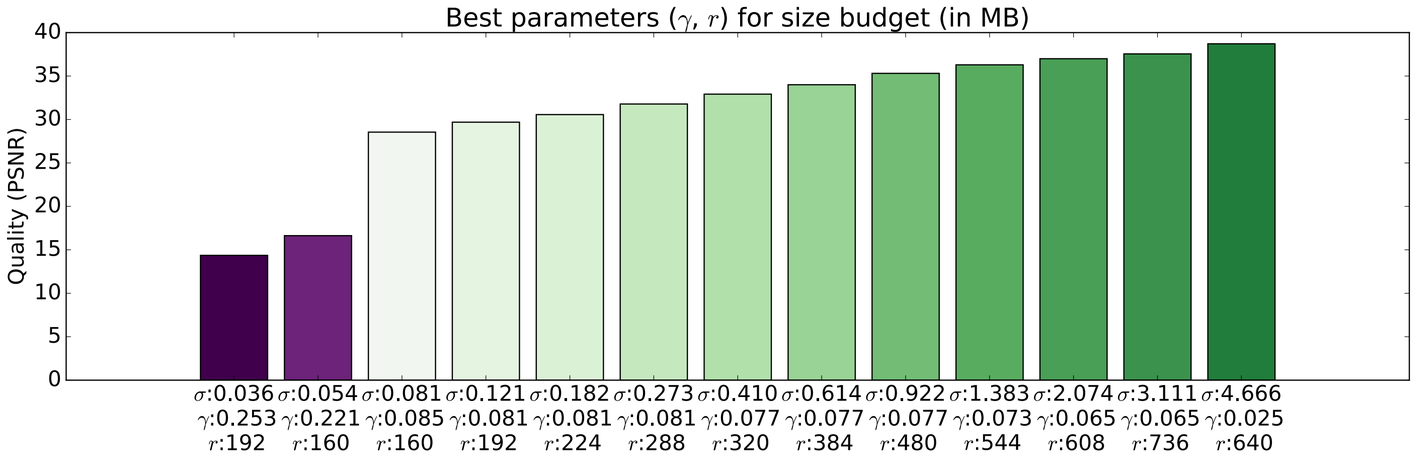
0.081 MB
γ: 0.085
r: 160

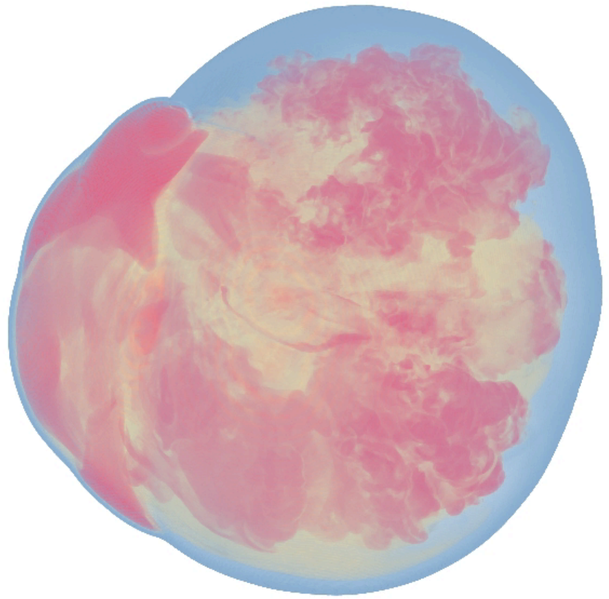


0.41 MB
γ: 0.077
r: 320

2.074 MB
γ: 0.065
r: 608

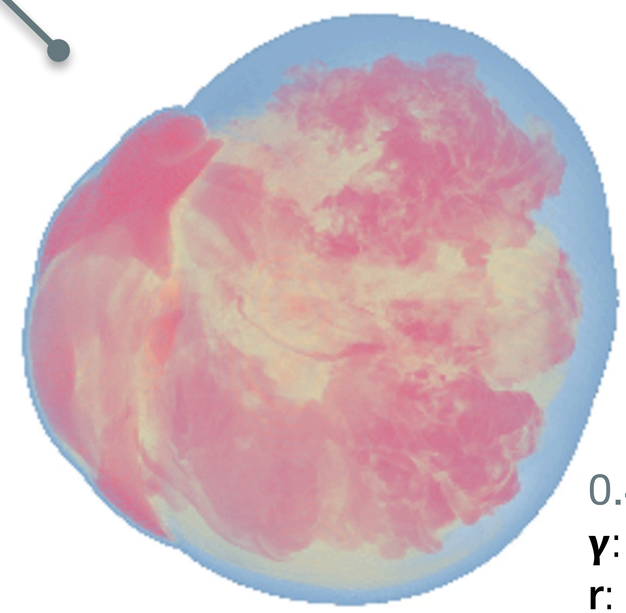
Auto-Tuning - Supernova 20





ref

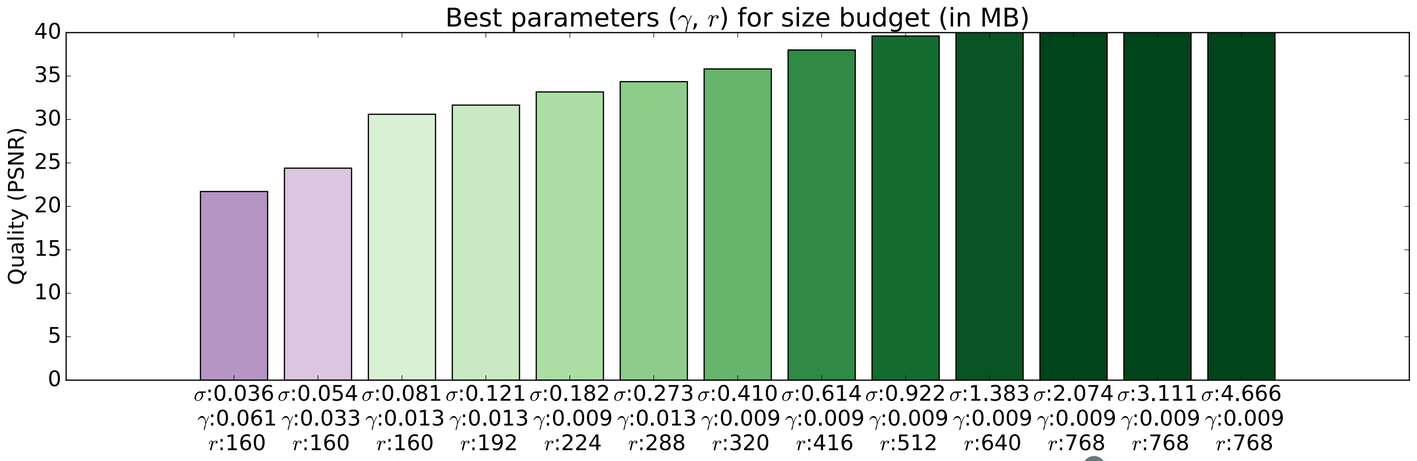
0.081 MB
γ: 0.085
r: 160

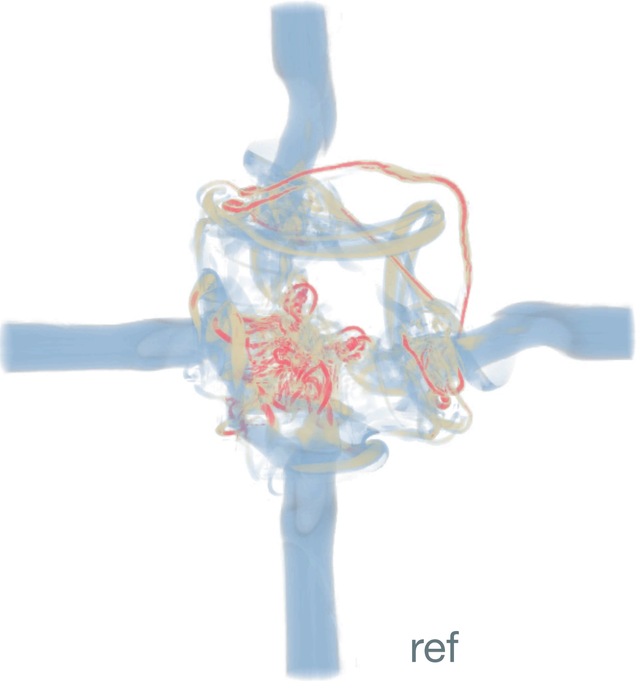


0.41 MB
γ: 0.077
r: 320

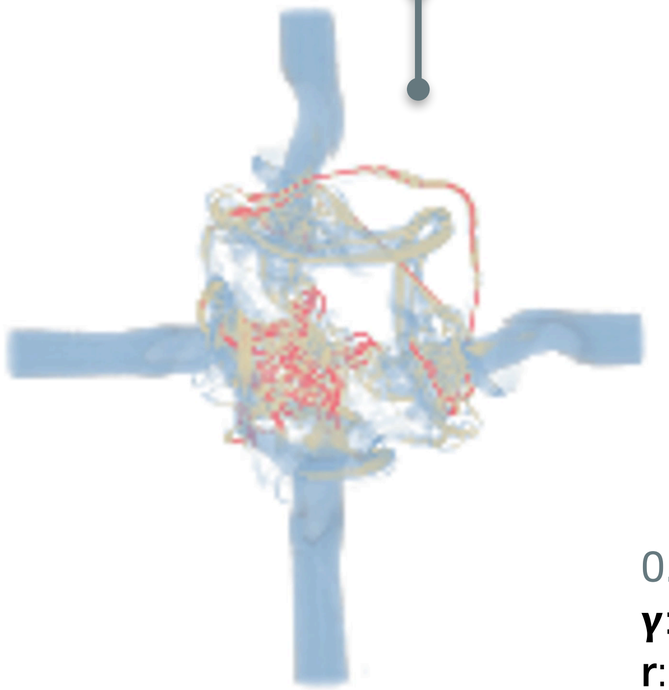
2.074 MB
γ: 0.065
r: 608

Auto-Tuning - λ_2

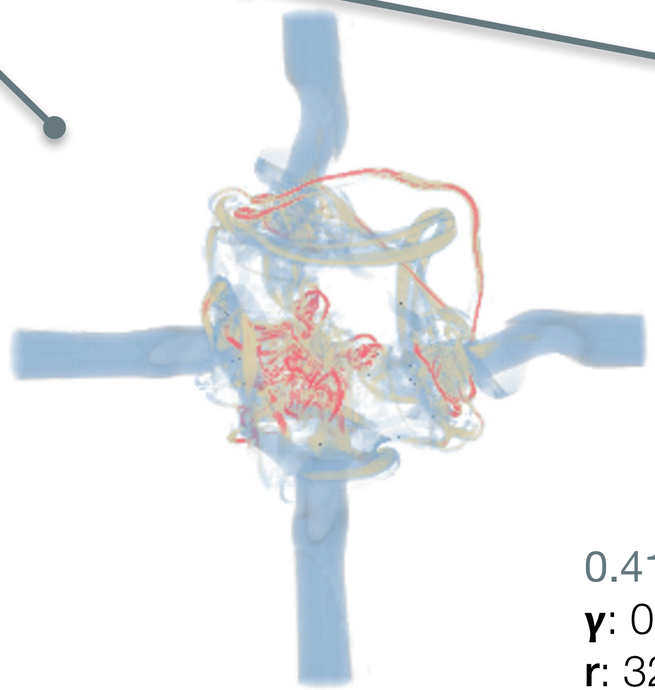




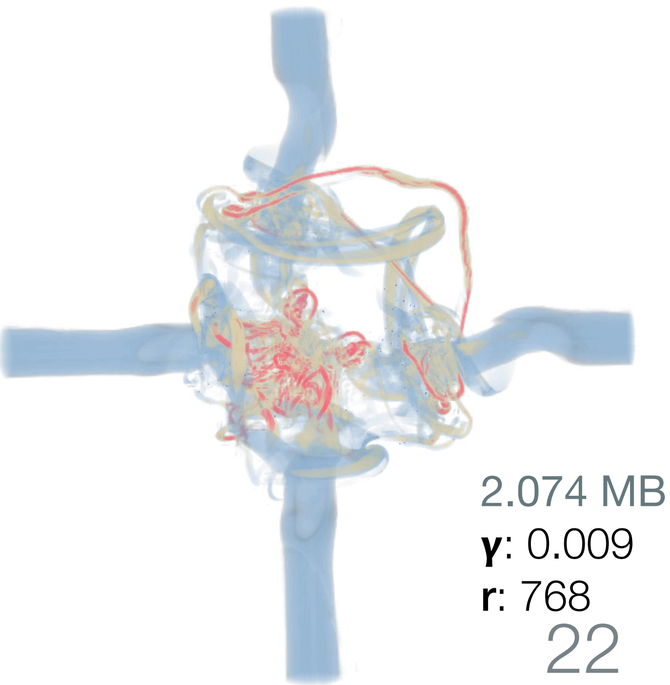
ref



0.081 MB
γ: 0.013
r: 160

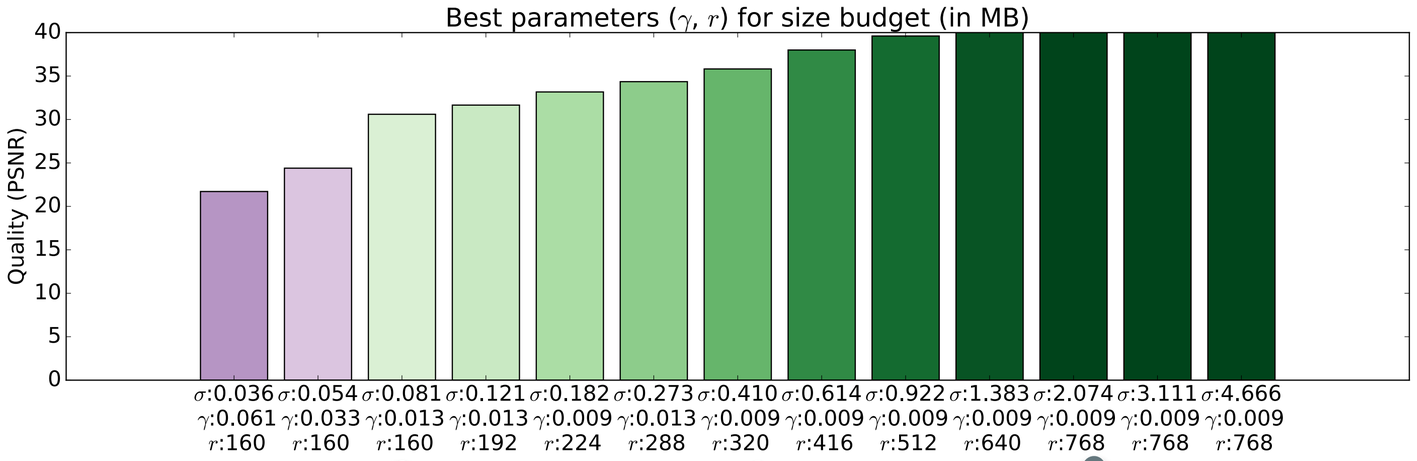


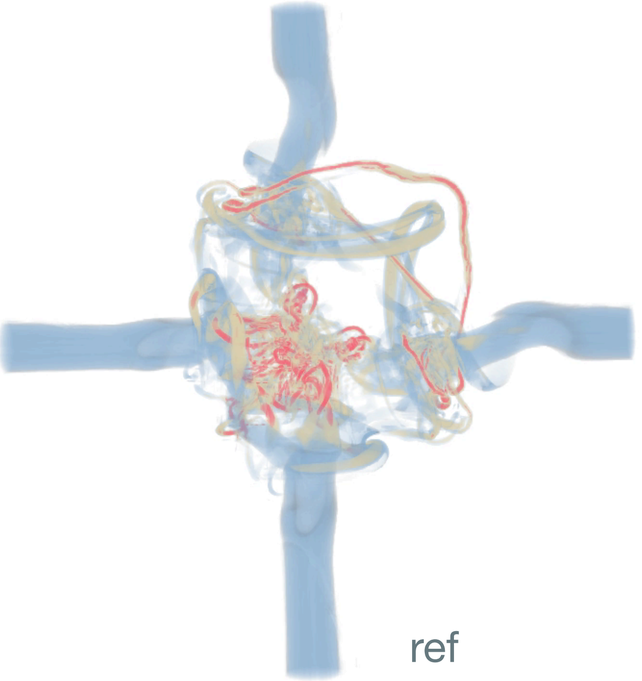
0.41 MB
γ: 0.009
r: 320



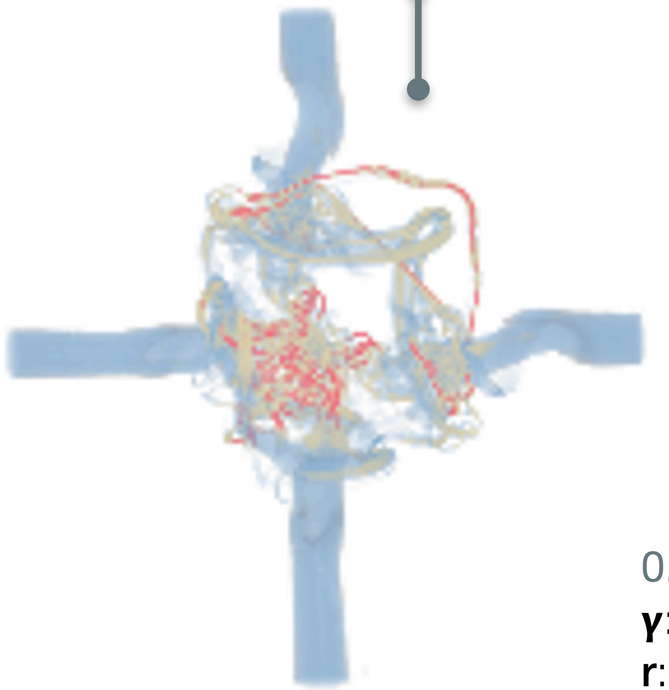
2.074 MB
γ: 0.009
r: 768

Auto-Tuning - λ_2

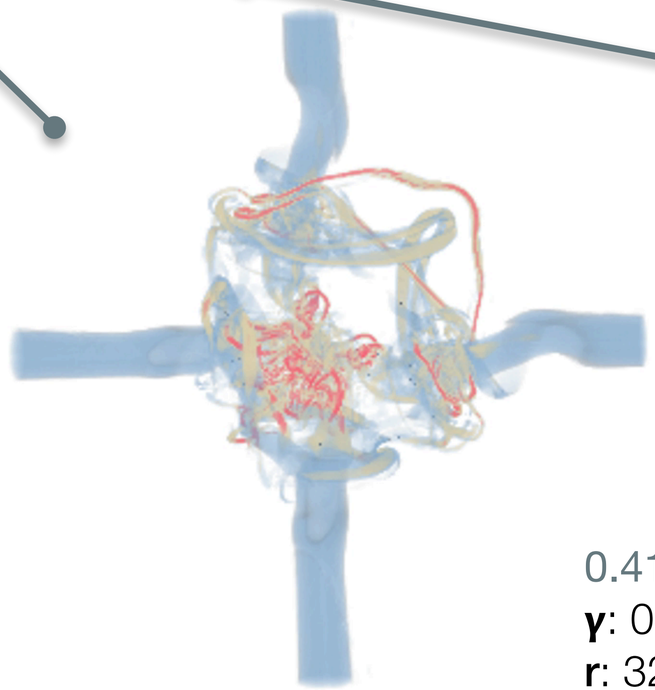




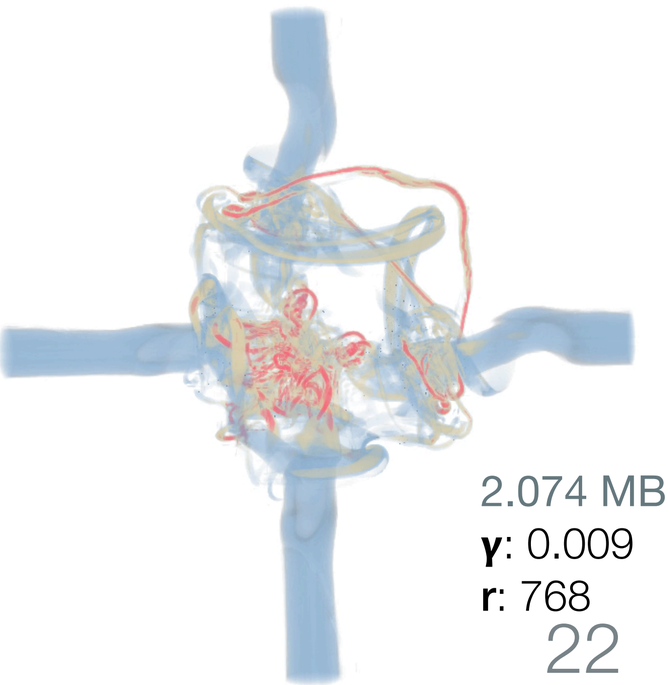
ref



0.081 MB
γ: 0.013
r: 160

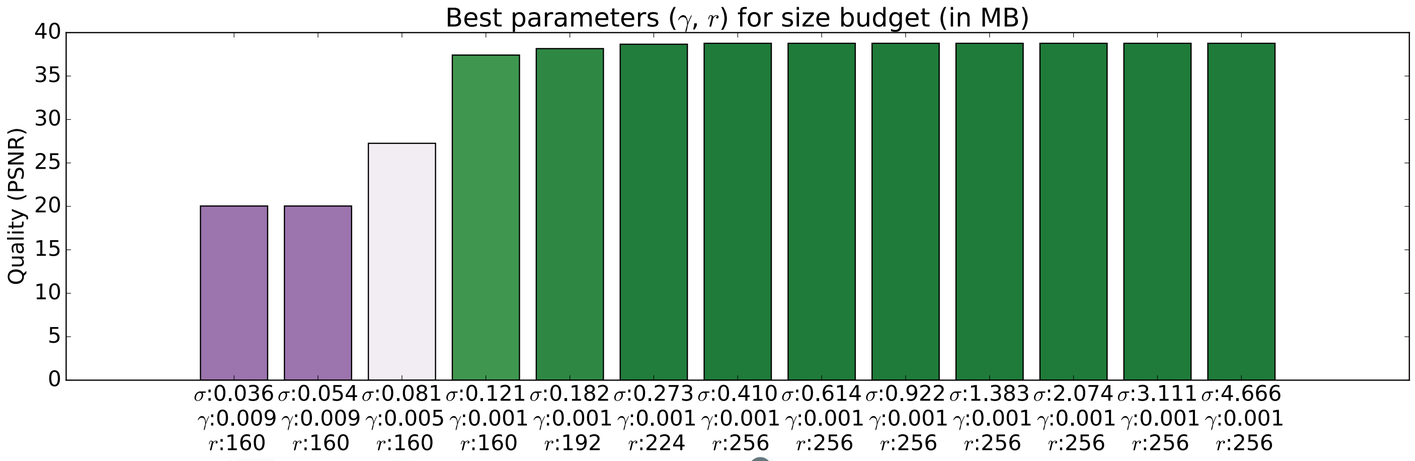


0.41 MB
γ: 0.009
r: 320

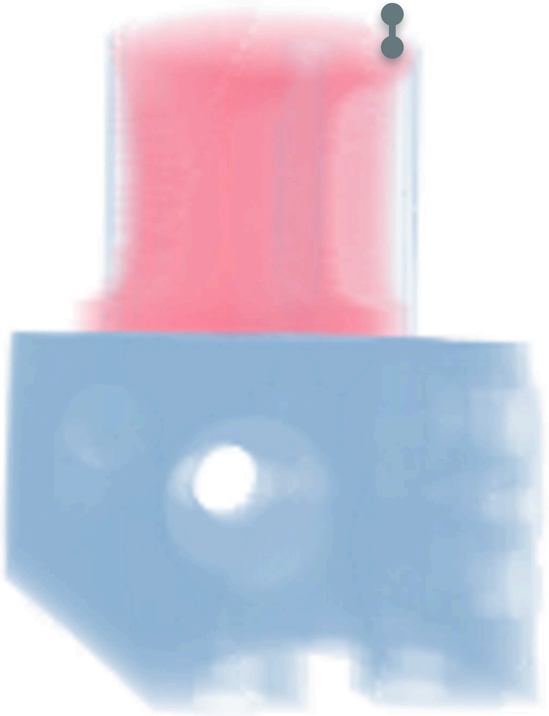


2.074 MB
γ: 0.009
r: 768

Auto-Tuning - Zeiss



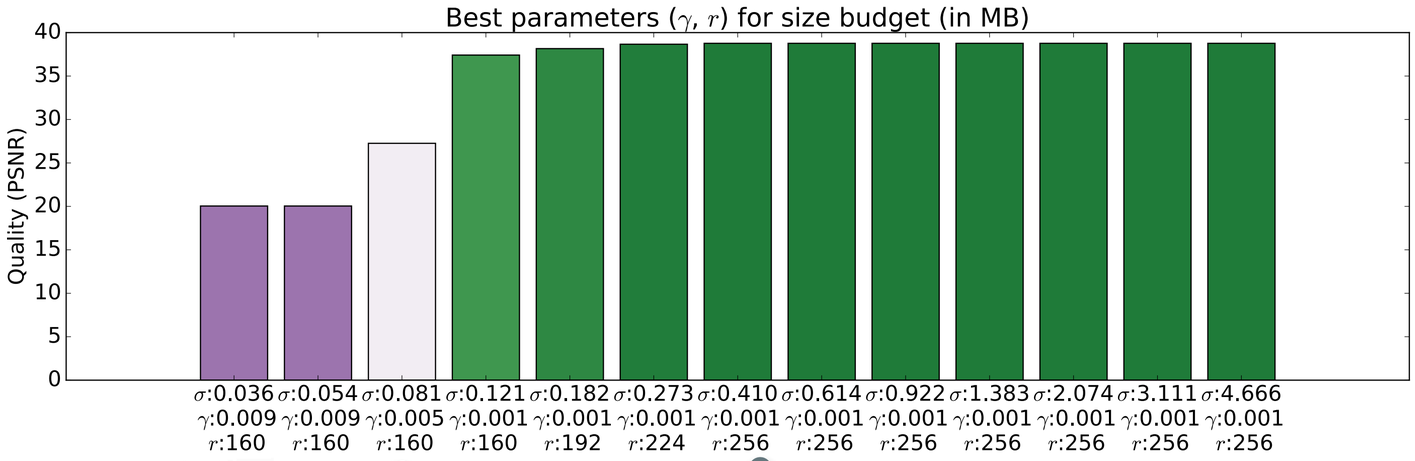
ref



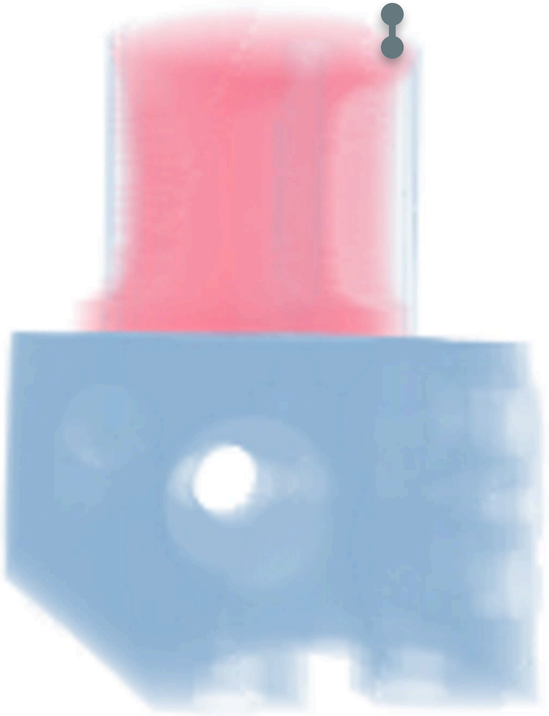
0.081 MB
γ: 0.005
r: 160

0.41 MB
γ: 0.001
r: 256

Auto-Tuning - Zeiss



ref



0.081 MB
γ: 0.005
r: 160

0.41 MB
γ: 0.001
r: 256

Transferability of Tuning Results

- tuning results for series of time steps
  - two supernova time steps (20 and 40)
  - apply tuning result $\mathbf{P_M} = (\mathbf{r}, \boldsymbol{\gamma})$ to respective other time step

- parameter settings translate to a certain extent
  - yet no guarantees can be given without explicit consideration
  - target limit exceed by up ≈ 50% …
    - when going from less complex 40 to 20
  - generally depends on similarity of characteristics

- when optimizing for a time series
  - a couple of characteristic time steps should be picked
  - simplest approach: use most conservative parameter settings for requested bounds

source	γ	r	σ	q	τ
$\sigma_{\text{target}} = 0.05$ MB					
40 (from 40)	0.245	160.0	0.053	21.911	0.077
20 (from 40)	0.245	160.0	0.044	16.316	0.075
40 (from 20)	0.221	160.0	0.064	25.695	0.077
20 (from 20)	0.221	160.0	0.052	16.633	0.078
$\sigma_{\text{target}} = 0.08$ MB					
40 (from 40)	0.077	160.0	0.075	27.777	0.078
20 (from 40)	0.077	160.0	0.100	28.915	0.077
40 (from 20)	0.085	160.0	0.072	27.617	0.078
20 (from 20)	0.085	160.0	0.077	28.546	0.076
$\sigma_{\text{target}} = 0.12$ MB					
40 (from 40)	0.073	192.0	0.109	28.845	0.098
20 (from 40)	0.073	192.0	0.169	30.026	0.099
40 (from 20)	0.081	192.0	0.106	28.731	0.098
20 (from 20)	0.081	192.0	0.117	29.691	0.099
$\sigma_{\text{target}} = 0.18$ MB					
40 (from 40)	0.149	256.0	0.165	30.474	0.154
20 (from 40)	0.149	256.0	0.159	29.656	0.155
40 (from 20)	0.081	224.0	0.144	29.724	0.124
20 (from 20)	0.081	224.0	0.159	30.565	0.125
$\sigma_{\text{target}} = 0.27$ MB					
40 (from 40)	0.149	320.0	0.257	31.714	0.216
20 (from 40)	0.149	320.0	0.249	30.381	0.216
40 (from 20)	0.081	288.0	0.237	31.393	0.184
20 (from 20)	0.081	288.0	0.262	31.782	0.183
$\sigma_{\text{target}} = 0.41$ MB					
40 (from 40)	0.073	352.0	0.367	32.892	0.250
20 (from 40)	0.073	352.0	0.570	33.754	0.249
40 (from 20)	0.077	320.0	0.299	32.276	0.211
20 (from 20)	0.077	320.0	0.400	32.918	0.214
$\sigma_{\text{target}} = 0.61$ MB					
40 (from 40)	0.073	448.0	0.593	34.429	0.371
20 (from 40)	0.073	448.0	0.922	35.208	0.371
40 (from 20)	0.077	384.0	0.431	33.341	0.286
20 (from 20)	0.077	384.0	0.576	34.001	0.286
$\sigma_{\text{target}} = 0.92$ MB					
40 (from 40)	0.069	544.0	0.885	35.617	0.517
20 (from 40)	0.069	544.0	1.520	36.464	0.516
40 (from 20)	0.077	480.0	0.674	34.783	0.417
20 (from 20)	0.077	480.0	0.901	35.309	0.415

Limitations and Directions For Future Work

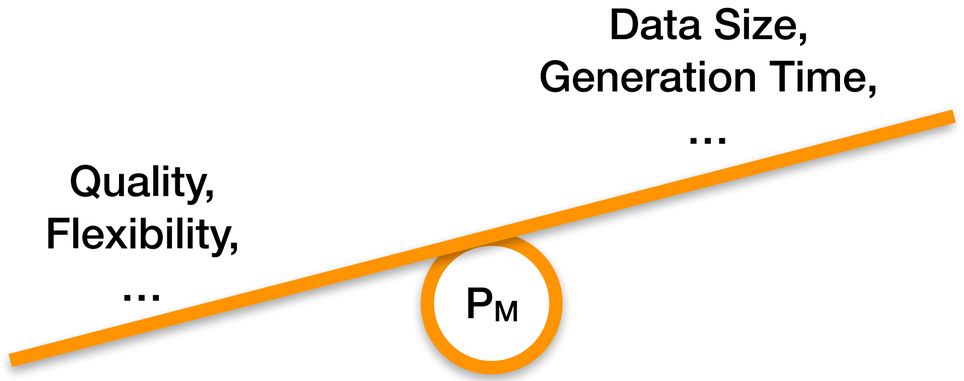
- IRs can be tuned toward certain goals …
  - … but no guarantees can be made this way
  - ➡ particularly for a time series: base on collection of characteristic time steps

- this work: storage space and computation time budgets considered
  - ➡ also consider other factors of interest (e.g., energy consumption)

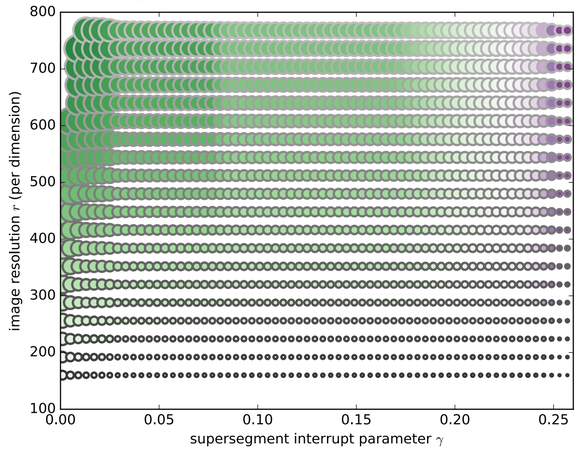
- even if IR is flexibly tunable, the range of achievable results is limited
  - ➡ auto-tuning could consider different IR, e.g., switch to sparse representations

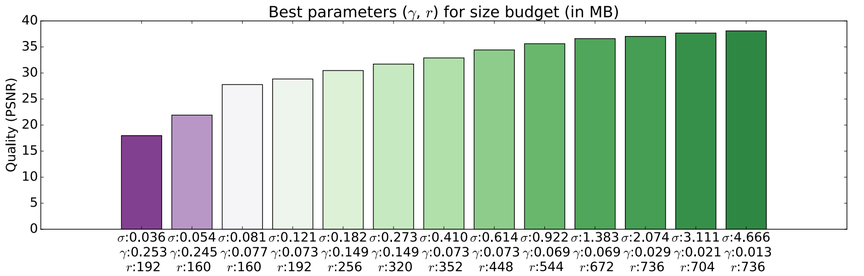
- this work: predefined set of PM (based on a priori experiments)
  - collection of results data can easily be distributed and cached in advance
  - ➡ impact of arbitrary utility functions $\boldsymbol{\nu}$ can be evaluated efficiently
    - fixed $\boldsymbol{\nu}$: integrated, adaptive scheme could be faster / more accurate

Conclusion

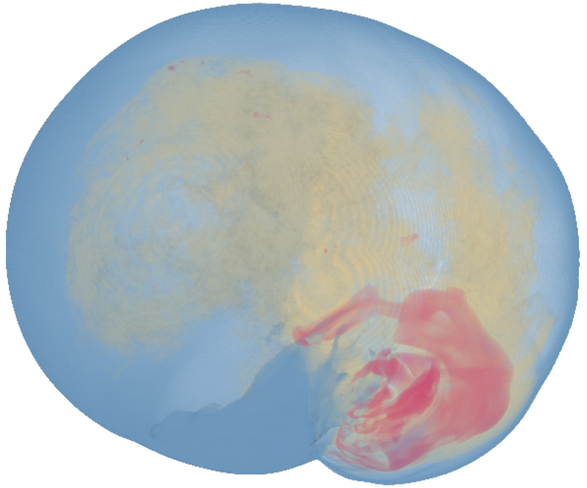


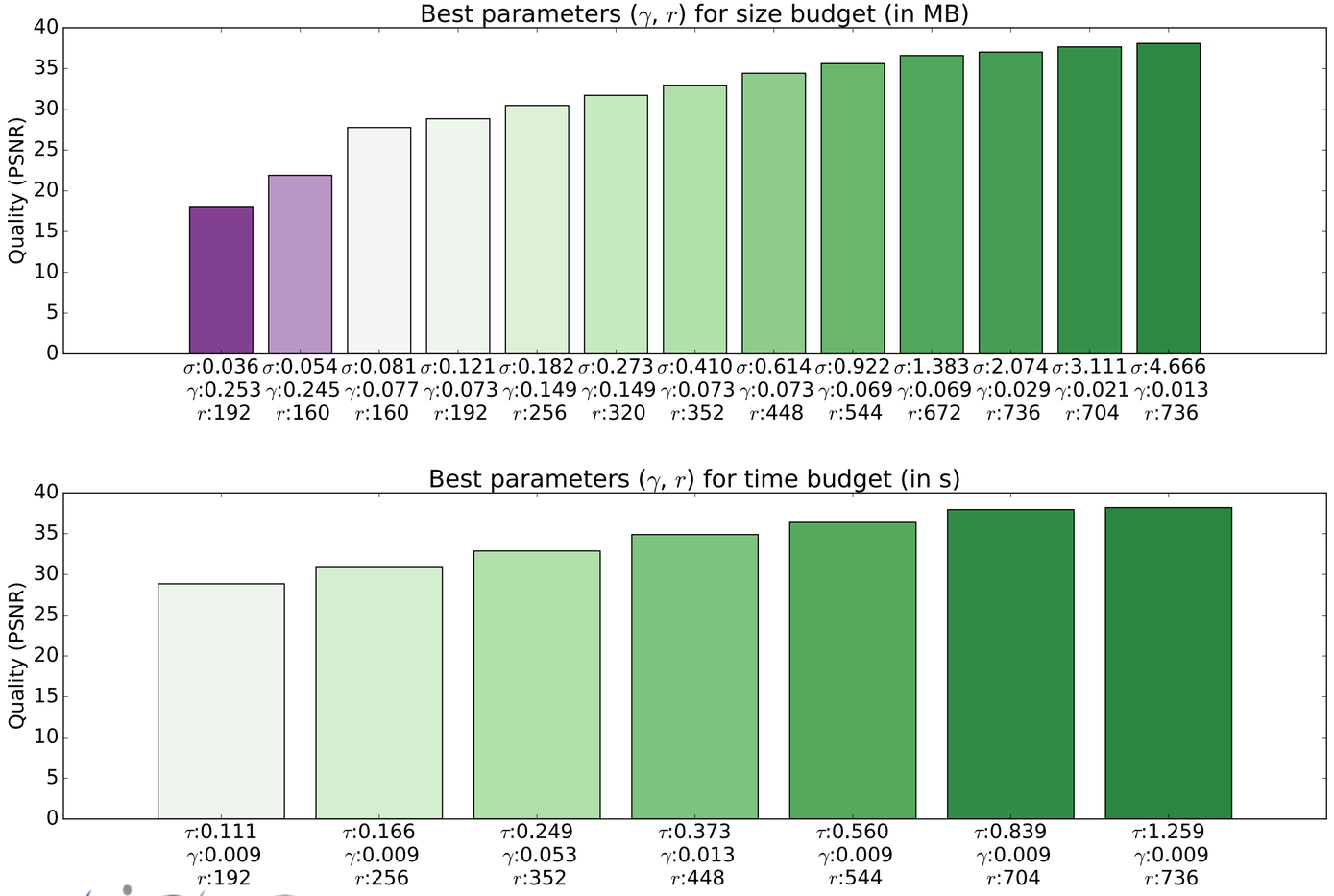
- IR for hybrid in-situ visualization
  - decrease data size
  - maintain flexibility for a posteriori exploration
- IR generation inherently involves a trade-off
  - tunable via parameters for most implementations of IR
- goal of this work: optimize generation of IR
  - analysis and quantification of the impact of IR parameters
  - auto-tuning for different constraints (like time or data size)
- many directions for future work
  - time series / ensembles, other factors of interest, switching between IRs, dynamic tuning, …
  - also consider other application domains beyond volume / scientific visualization





Thank You!




Best parameters (γ, r) for size budget (in MB)
Quality (PSNR)
0
5
10
15
20
25
30
35
40
σ:0.036 γ:0.253 r:192
σ:0.054 γ:0.245 r:160
σ:0.081 γ:0.077 r:160
σ:0.121 γ:0.073 r:192
σ:0.182 γ:0.149 r:256
σ:0.273 γ:0.149 r:320
σ:0.410 γ:0.073 r:352
σ:0.614 γ:0.073 r:448
σ:0.922 γ:0.069 r:544
σ:1.383 γ:0.069 r:672
σ:2.074 γ:0.029 r:736
σ:3.111 γ:0.021 r:704
σ:4.666 γ:0.013 r:736

Best parameters (γ, r) for time budget (in s)
Quality (PSNR)
0
5
10
15
20
25
30
35
40
τ:0.111 γ:0.009 r:192
τ:0.166 γ:0.009 r:256
τ:0.249 γ:0.053 r:352
τ:0.373 γ:0.013 r:448
τ:0.560 γ:0.009 r:544
τ:0.839 γ:0.009 r:704
τ:1.259 γ:0.009 r:736

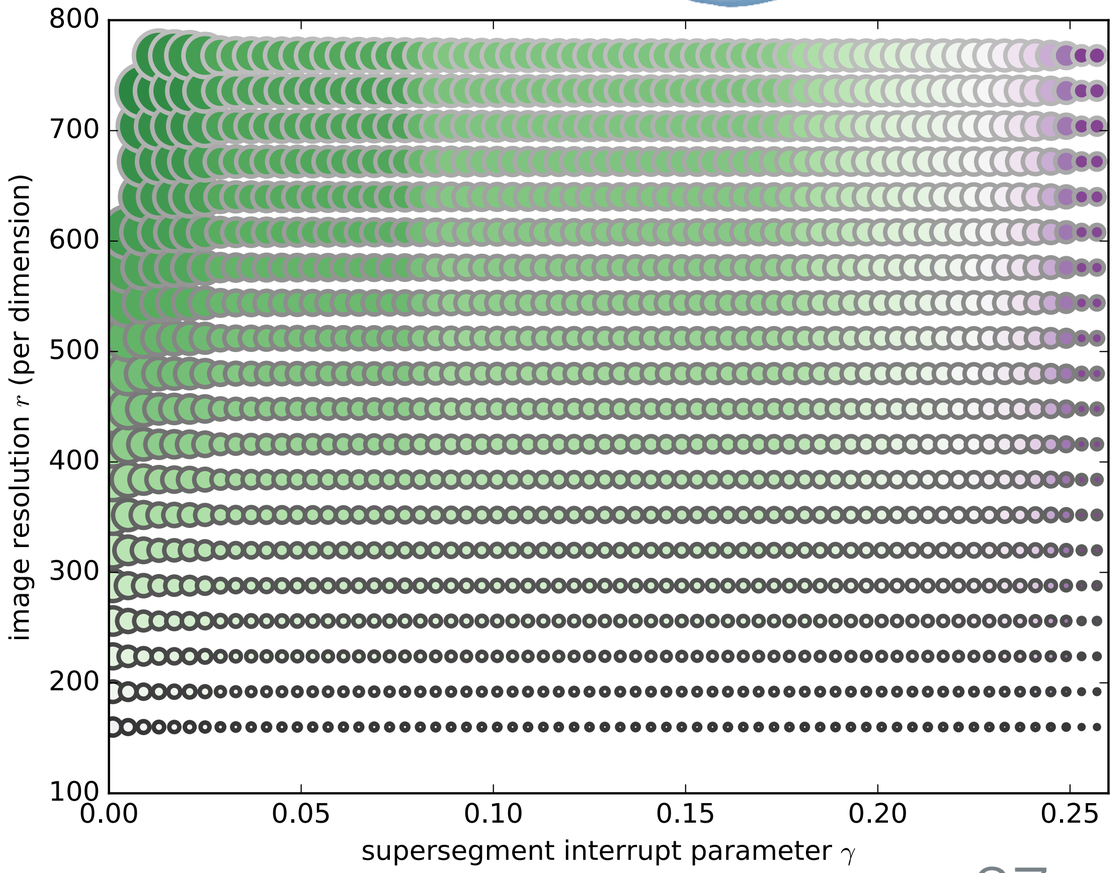


image resolution r (per dimension)
100
200
300
400
500
600
700
800
0.00
0.05
0.10
0.15
0.20
0.25
supersegment interrupt parameter γ