Fast Approximate Energy Minimization via Graph Cuts

Felix Wechsler

Computer Aided Medical Procedures
Technische Universität München

November 15, 2017

Outline

Used paper: Boykov, Veksler, and Zabih, "Fast Approximate Energy Minimization via Graph Cuts"

Basics

Standard CV Problem

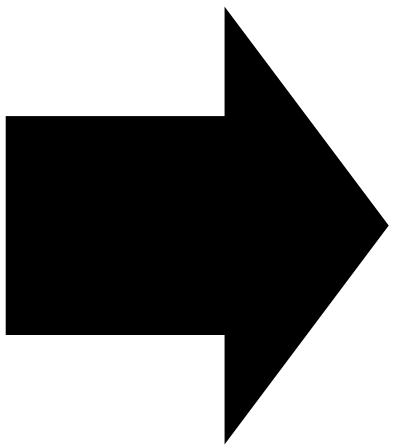
- Tackling noisy images
- Piecewise smooth variation of intensity

$\Longrightarrow$ energy minimization

Standard CV Problem

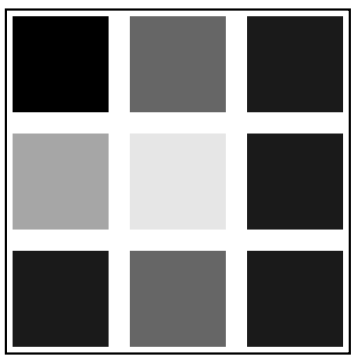
- Tackling noisy images
- Piecewise smooth variation of intensity

$\Longrightarrow$ energy minimization



Labeling

- Every pixel $p \in \mathcal{P}$ has label $f_p \in \mathcal{L}$
- Labeling of image is called f
- Label is intensity in image restoration



labeling f

Labeling

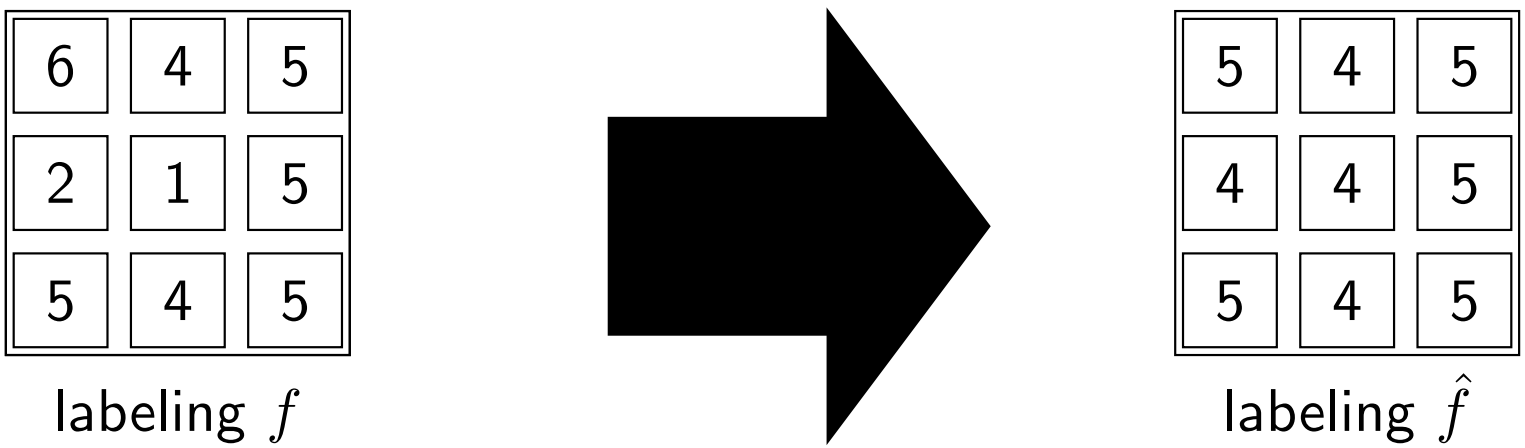
- Every pixel $p \in \mathcal{P}$ has label $f_p \in \mathcal{L}$
- Labeling of image is called f
- Label is intensity in image restoration

6	4	5
2	1	5
5	4	5

labeling f

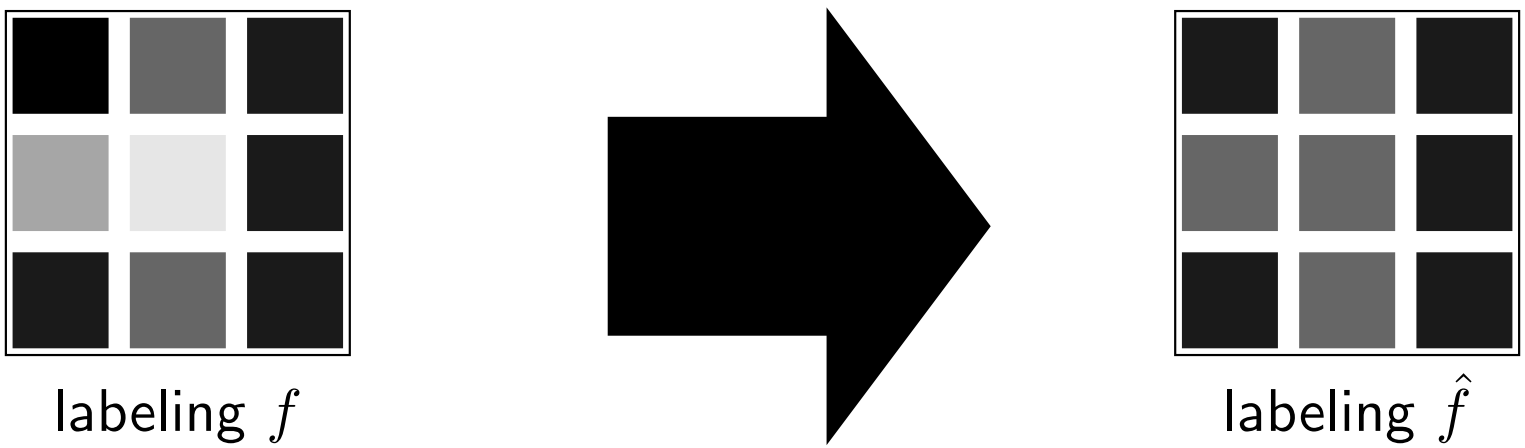
Labeling

- Every pixel $p \in \mathcal{P}$ has label $f_p \in \mathcal{L}$
- Labeling of image is called f
- Label is intensity in image restoration



Labeling

- Every pixel $p \in \mathcal{P}$ has label $f_p \in \mathcal{L}$
- Labeling of image is called f
- Label is intensity in image restoration



labeling f labeling $\hat{f}$

Energy Function

$$E(f) = E_{\text{smooth}}(f) + E_{\text{data}}(f) \tag{1}$$

- $E_{\text{smooth}} \mathrel{\hat{=}}$ smoothness of image
- $E_{\text{data}} \mathrel{\hat{=}}$ disagreement between f and image data

$$E_{\text{data}} = \sum_{p \in \mathcal{P}} D_p(f_p) \tag{2}$$

- i.e.: $D_p(f_p) = (f_p - i_p)^2$

  f_p label pixel p

  i_p original intensity of pixel p

Smoothness Term

$$E_{\text{smooth}} = \sum_{\{p,q\} \in \mathcal{N}} V_{p,q}(f_p, f_q) \tag{3}$$

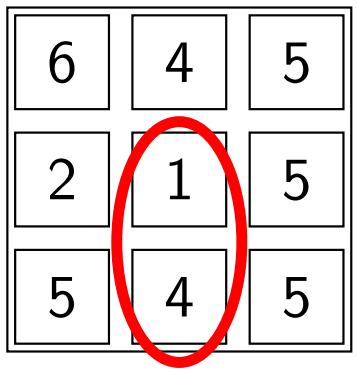
- $V_{p,q}$: potential function
- $\mathcal{N}$: set of pair of adjacent pixels

6	4	5
2	1	5
5	4	5

Smoothness Term

$$E_{\text{smooth}} = \sum_{\{p,q\} \in \mathcal{N}} V_{p,q}(f_p, f_q) \tag{3}$$

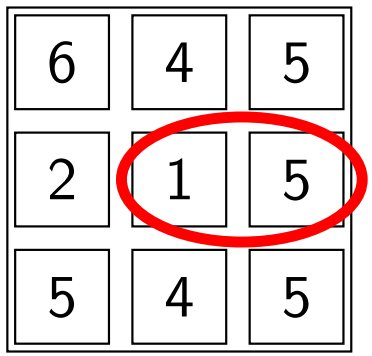
- $V_{p,q}$: potential function
- $\mathcal{N}$: set of pair of adjacent pixels



Smoothness Term

$$E_{\text{smooth}} = \sum_{\{p,q\} \in \mathcal{N}} V_{p,q}(f_p, f_q) \tag{3}$$

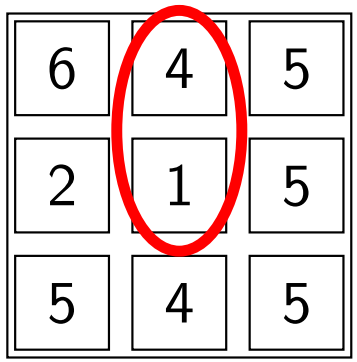
- $V_{p,q}$: potential function
- $\mathcal{N}$: set of pair of adjacent pixels



Smoothness Term

$$E_{\text{smooth}} = \sum_{\{p,q\} \in \mathcal{N}} V_{p,q}(f_p, f_q) \tag{3}$$

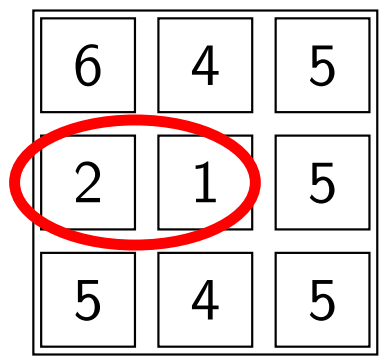
- $V_{p,q}$: potential function
- $\mathcal{N}$: set of pair of adjacent pixels



Smoothness Term

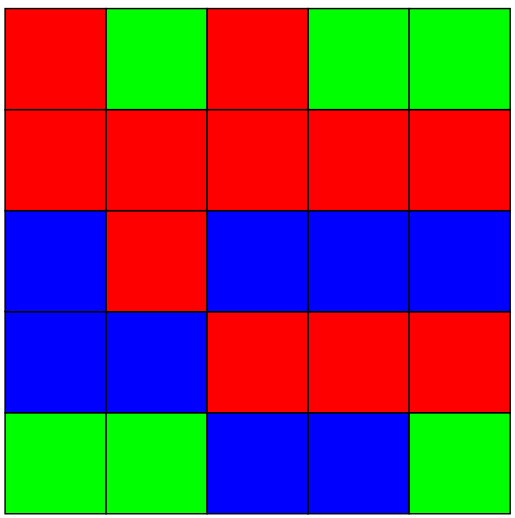
$$E_{\text{smooth}} = \sum_{\{p,q\} \in \mathcal{N}} V_{p,q}(f_p, f_q) \tag{3}$$

- $V_{p,q}$: potential function
- $\mathcal{N}$: set of pair of adjacent pixels



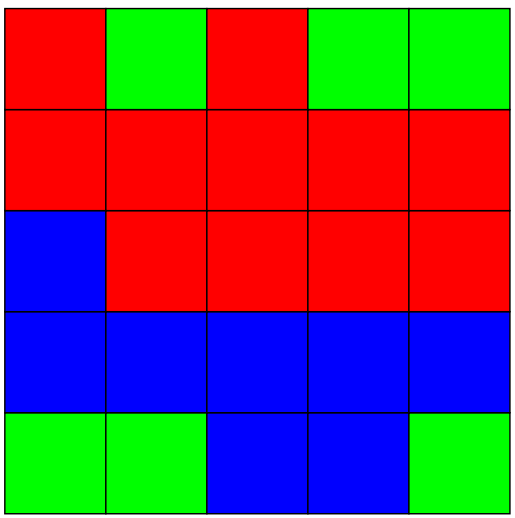
α-β-swap

- $\alpha \hat{=}$red and $\beta \hat{=}$blue
- $\mathcal{P}_{\alpha\beta}$ is set of all pixels with label α or β



Original Image

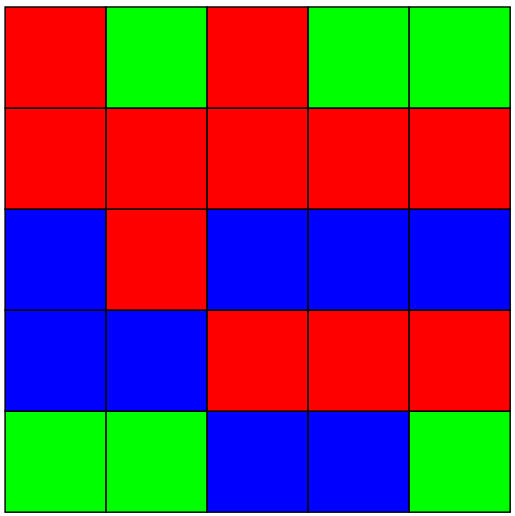
One α-β-swap



Optimized Image

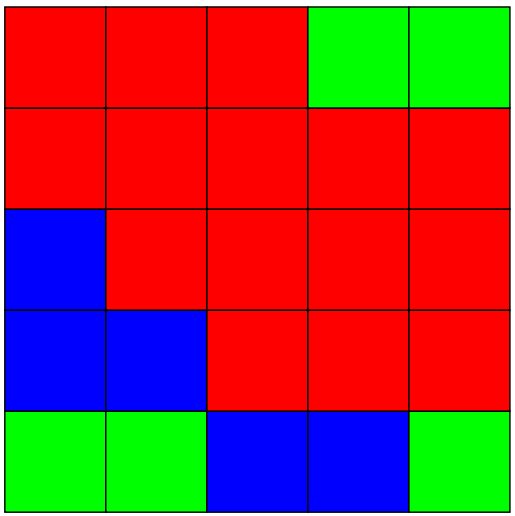
α-expansion

- $\alpha \hat{=}$red
- find optimum by expanding label α to other pixels



Original Image

One α-expansion



Optimized Image

Swap Algorithm

Swap Algorithm

1. Start with arbitrary labeling
2. success $:= 0$
3. for all $\{\alpha, \beta\} \subset \mathcal{L}$
   - 3.1 find $\hat{f} = \arg\min E(f')$ within one α-β-swap
   - 3.2 if $E(\hat{f}) < E(f)$ then $f = \hat{f}$
4. if success $== 1$ then goto 2
5. return f

Properties of Swap Algorithm

- complexity is in one cycle $\mathcal{O}(|\mathcal{L}|^2)$
- termination is in $\mathcal{O}(|\mathcal{P}|)$ cycles
- most of the improvements occur in first cycle
- finds local minimum

Graph Cutting

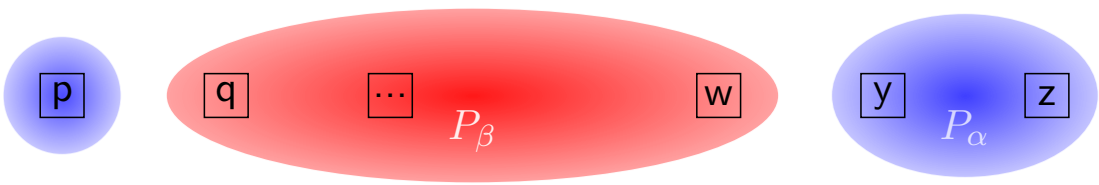
Construction of Graph

Construction of Graph

- Introduce pixels

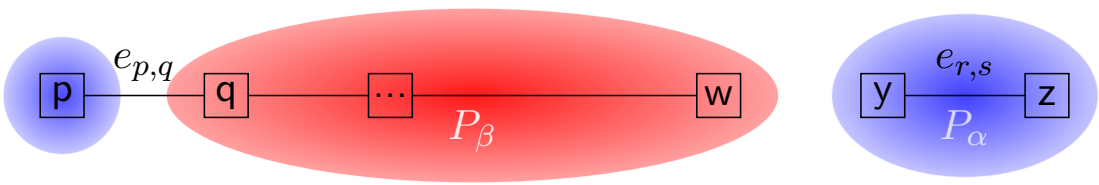
p q ... w y z

Construction of Graph



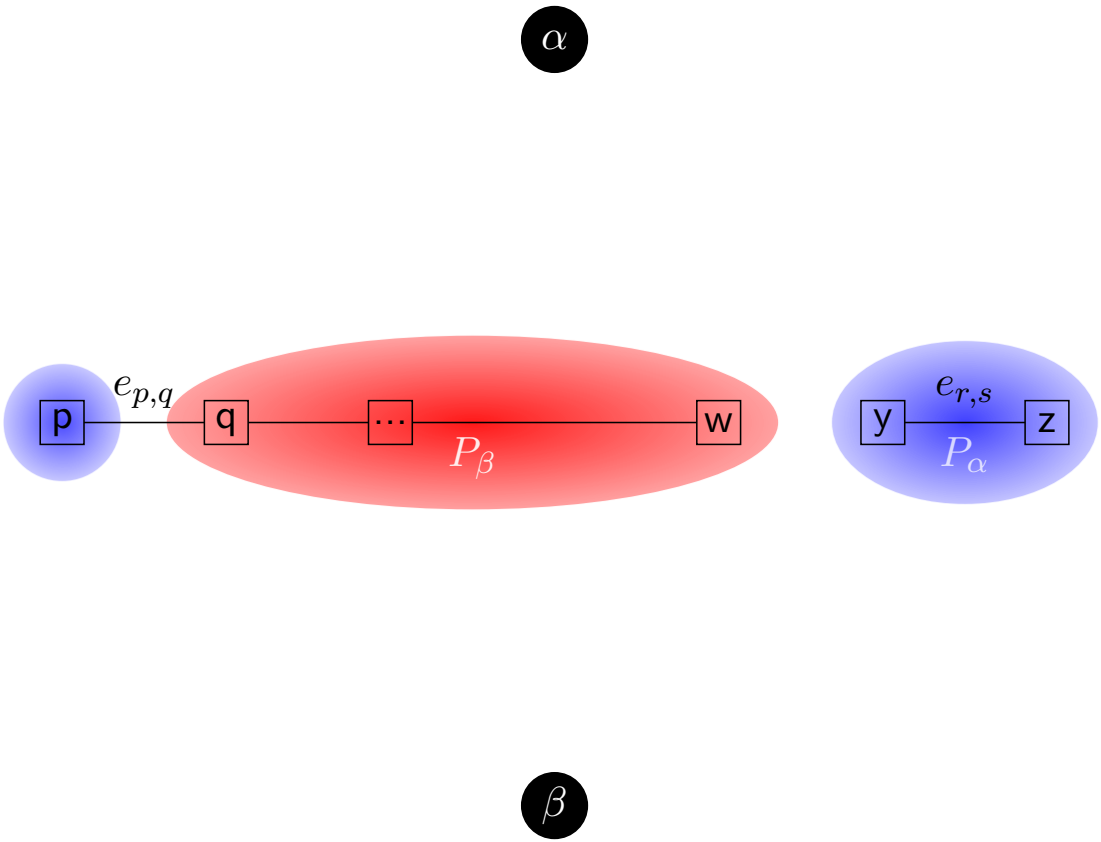
- Introduce pixels

Construction of Graph



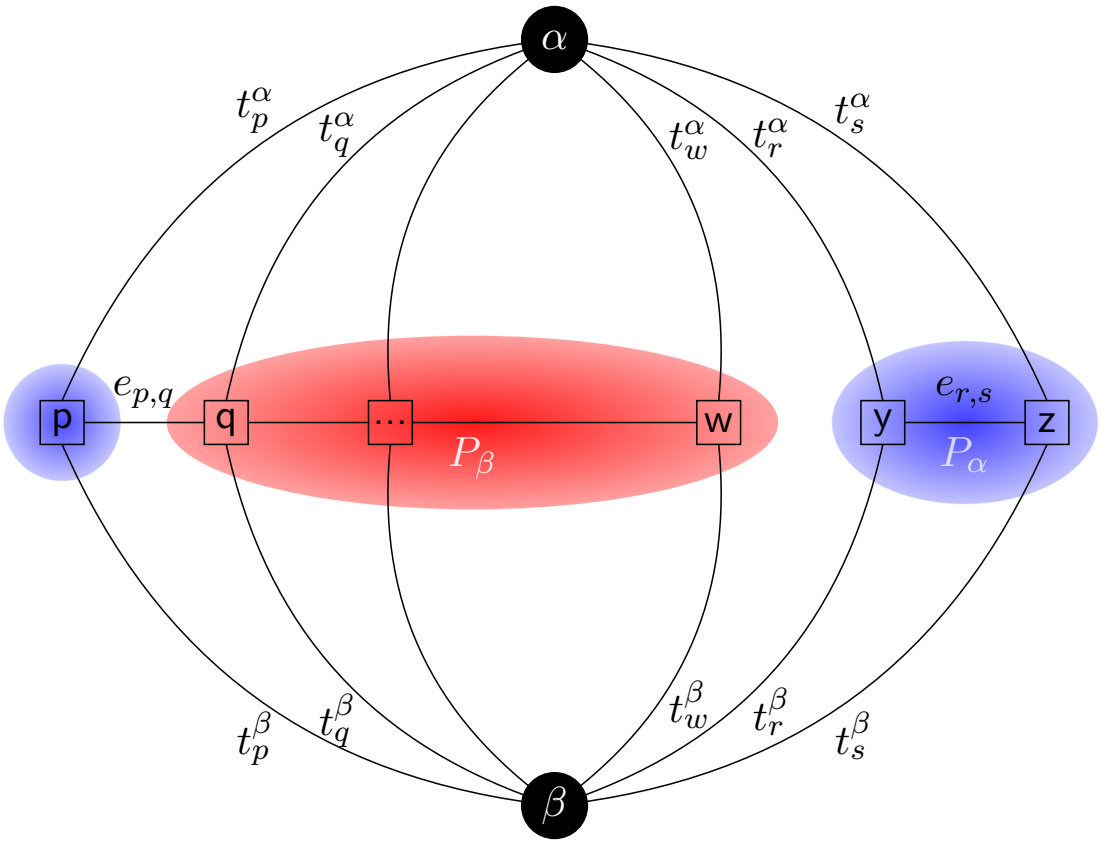
- Introduce pixels
- Connect neighbours

Construction of Graph



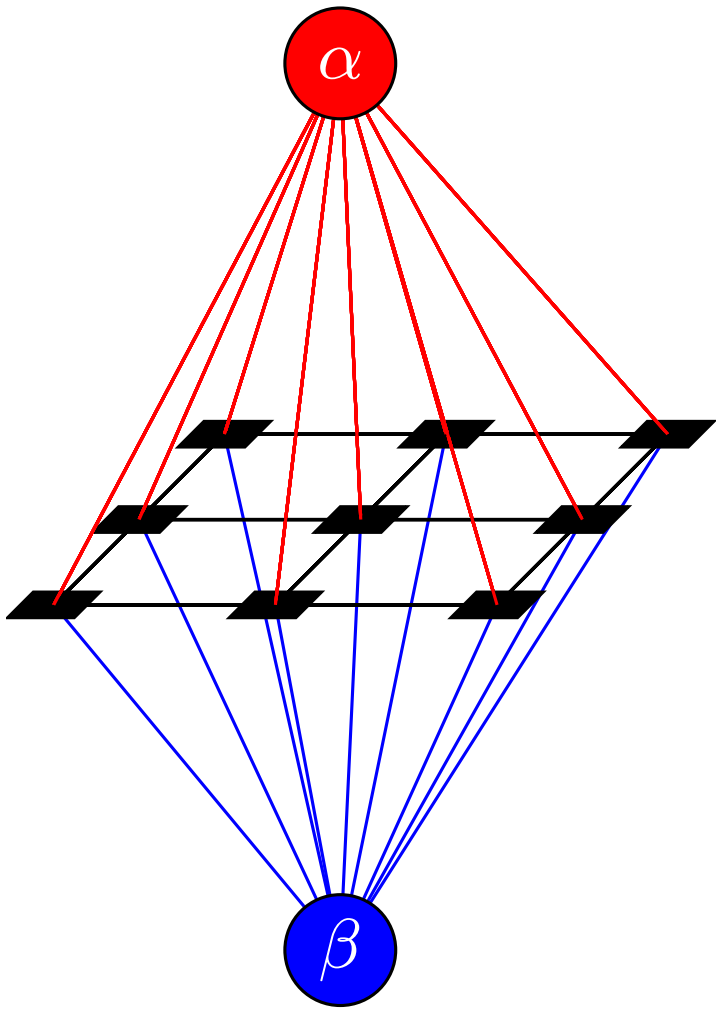
- Introduce pixels
- Connect neighbours
- Introduce two terminal nodes

Construction of Graph



- Introduce pixels
- Connect neighbours
- Introduce two terminal nodes
- Connect terminals and pixels

Construction of Graph



- Introduce pixels
- Connect neighbours
- Introduce two terminal nodes
- Connect terminals and pixels

Edge Weights

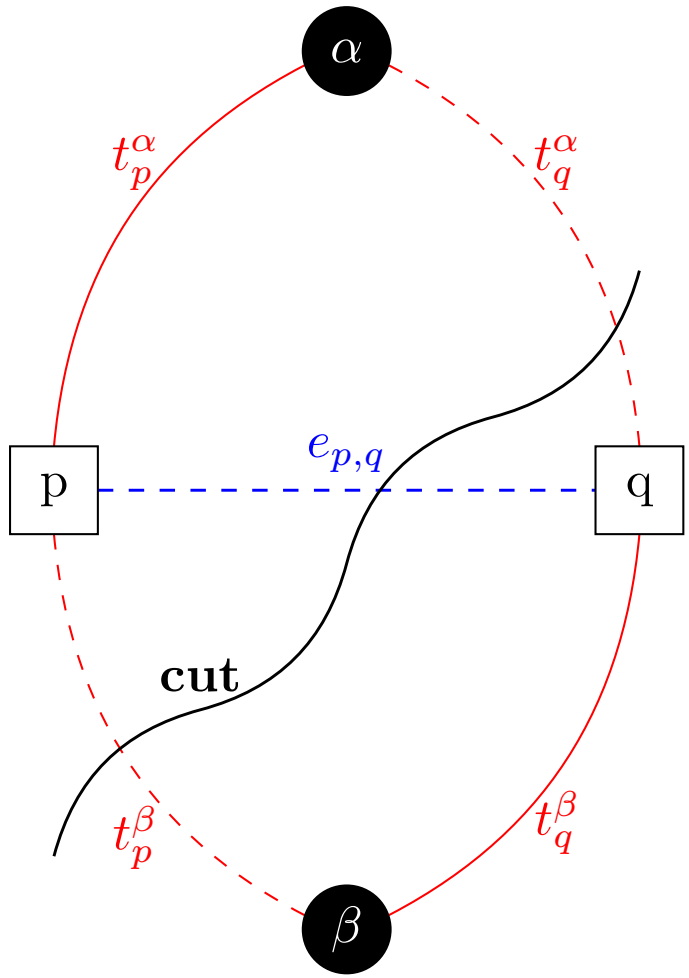
edge	weight	for
t_p^α	$D_p(\alpha) + \sum_{\substack{q \in \mathcal{N}_p \\ q \notin \mathcal{P}_{\alpha,\beta}}} V_{p,q}(\alpha, f_q)$	$p \in \mathcal{P}_{\alpha,\beta}$
t_p^β	$D_p(\beta) + \sum_{\substack{q \in \mathcal{N}_p \\ q \notin \mathcal{P}_{\alpha,\beta}}} V_{p,q}(\beta, f_q)$	$p \in \mathcal{P}_{\alpha,\beta}$
$e_{p,q}$	$V_{p,q}(\alpha, \beta)$	$\{p,q\} \in \mathcal{N} \wedge p, q \in \mathcal{P}_{\alpha,\beta}$

Possible Cuts

- Find $\hat{f} = \arg\min E(f')$ within one α-β-swap

$\rightarrow$ do minimal graph cut

$\rightarrow$ $f_p = \beta$ and $f_q = \alpha$



Minimum Cut

Cost of Minimut cut:

$$|\mathcal{C}| = \sum_{p \in \mathcal{P}_{\alpha,\beta}} \left| \mathcal{C} \cap \{t_p^\alpha, t_p^\beta\} \right| + \sum_{\substack{\{p,q\} \in \mathcal{N} \\ \{p,q\} \subset \mathcal{P}_{\alpha,\beta}}} |\mathcal{C} \cap e_{p,q}| \tag{4}$$

which can be rewritten to:

$$|\mathcal{C}| = E(f^{\mathcal{C}}) - K \tag{5}$$

- $f^{\mathcal{C}}$ labeling after cut
- K is constant for all cuts $\mathcal{C} \mathrel{\hat{=}}$ independent of $\mathcal{C}$

$\Longrightarrow$ **Finding min $\mathcal{C}$ is equivalent to finding energy min**

Finding Minimum Cut

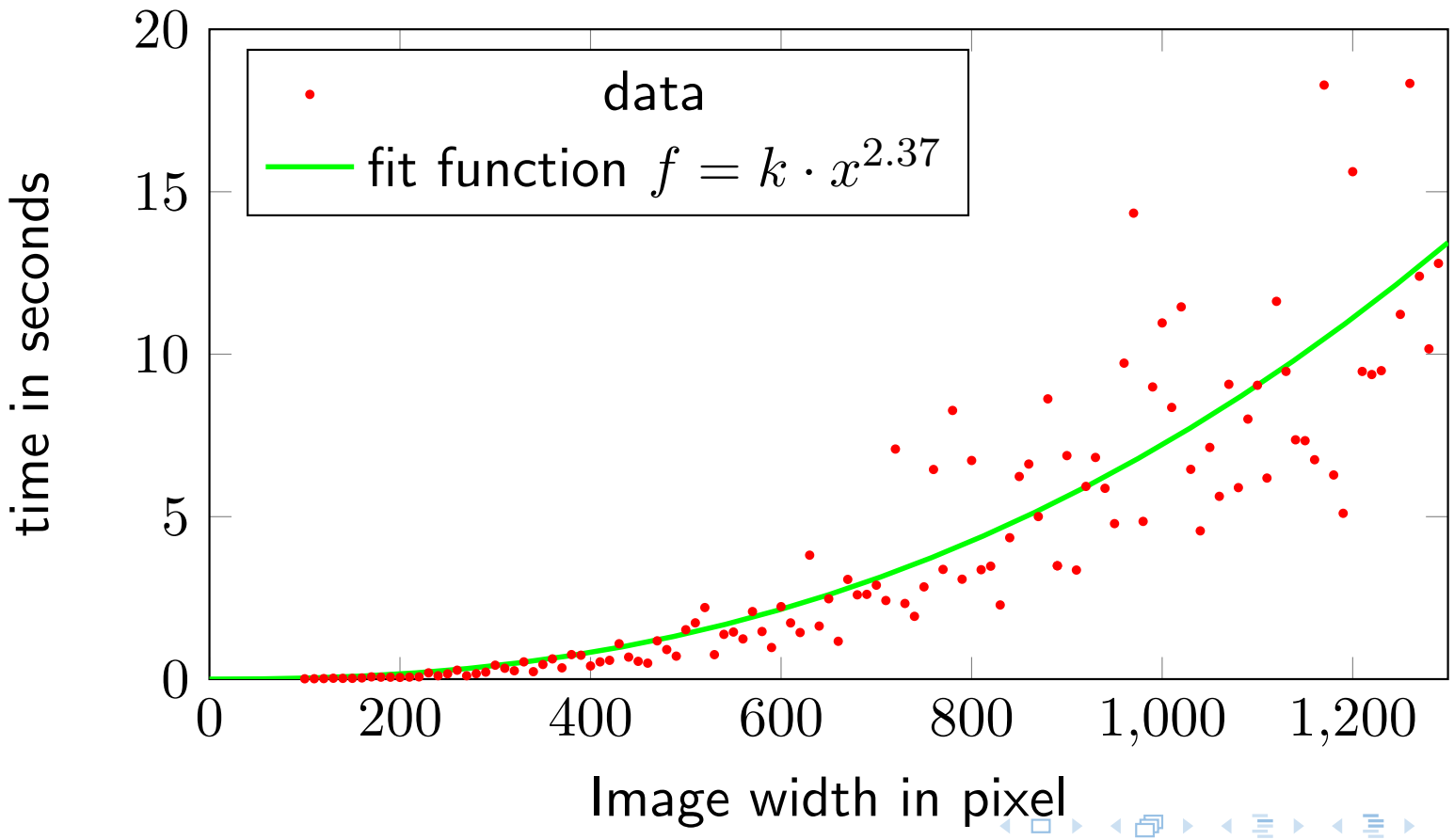
- famous Max-flow min-cut theorem
- Algorithms: *Push-relabel*-based or *Ford-Fulkerson*-based algorithms
- Kolmogorovs library `Maxflow`[1] is optimized for our graphs
- `PyMaxflow` is a python wrapper for `Maxflow` [2]

[1] Kolmogorov, *Maxflow*

[2] `http://pmneila.github.io/PyMaxflow/`

Scaling of graph-cut with `PyMaxflow`

`PyMaxflow` scales more than quadratic $\Longrightarrow$ confirms Boykov and Kolmogorov



Implementation

Own Implementation

- Implementation in Python available on `fcv.sumpi.org`
- uses `PyMaxflow`'s minimizing graph cut[3]
- objective: test different energy functions and images

[3] `http://pmneila.github.io/PyMaxflow/`

Energy Function

In paper used

$$D_p(f_p) = (f_p - i_p)^2 \tag{6}$$

$$V(f_p, f_q) = c \cdot |f_p - f_q| \tag{7}$$

but first is quadratic, second linear.

$\Longrightarrow$ another energy function which eliminates mismatch

$$\hat{D}_p(f_p) = \sqrt{|f_p^2 - i_p^2|} \tag{8}$$

f_p label of pixel p

i_p original intensity of pixel p

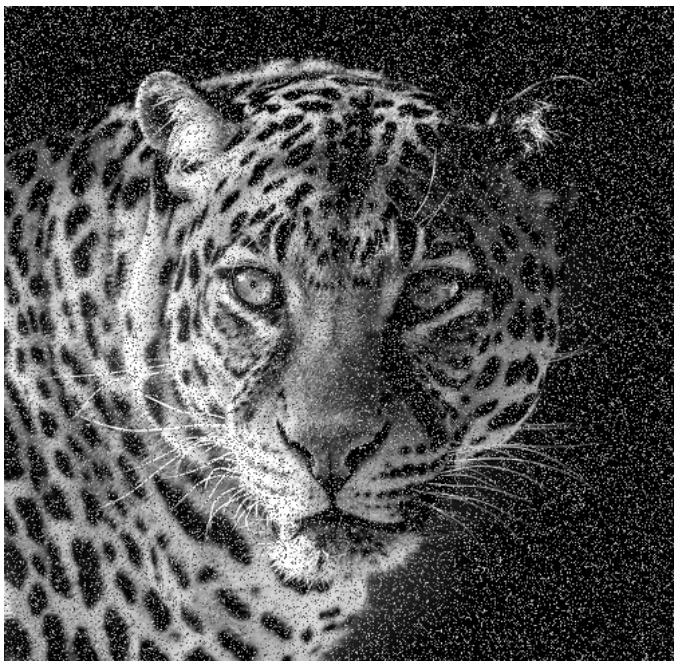
c some constant

Experiments

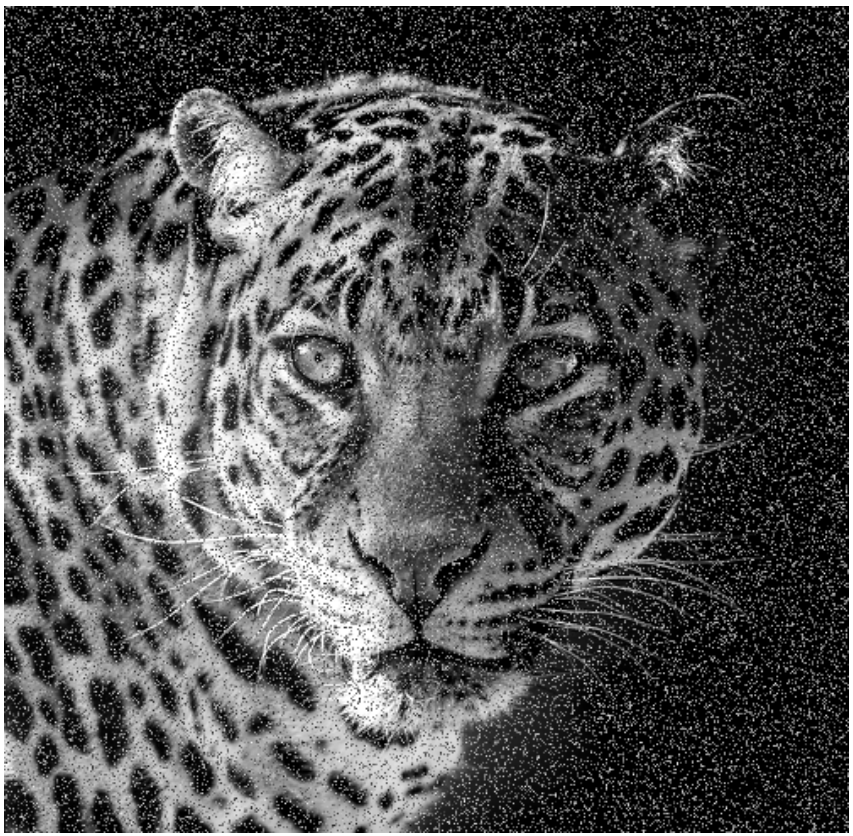
Denoising Leopard using $\hat{D}_p(f_p)$

Image with 500x489 pixels in grayscale:

20% pixels are randomly changed



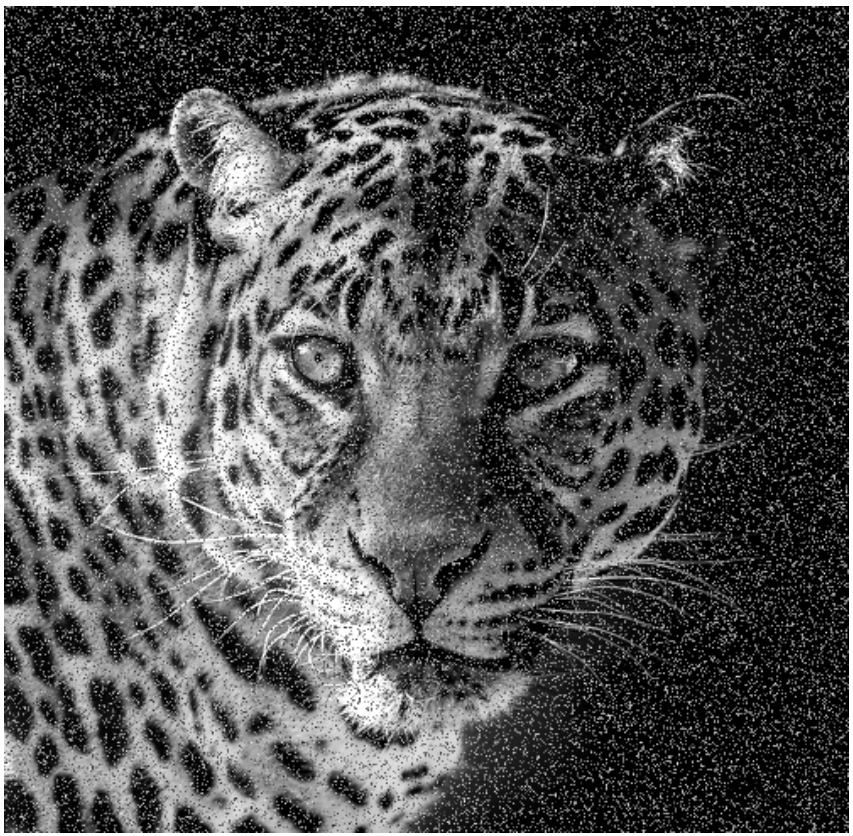
Denoising Leopard using $\hat{D}_p(f_p)$



Noisy leopard

Denoising Leopard using $\hat{D}_p(f_p)$

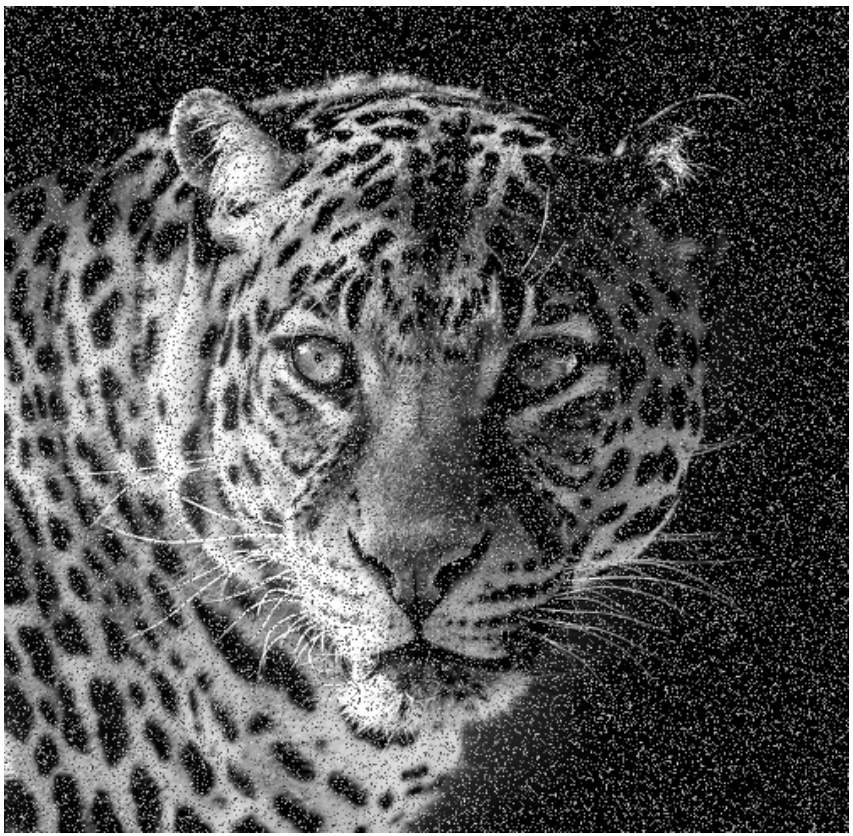
Denoised after 1 cycle



Noisy leopard

Denoising Leopard using $\hat{D}_p(f_p)$

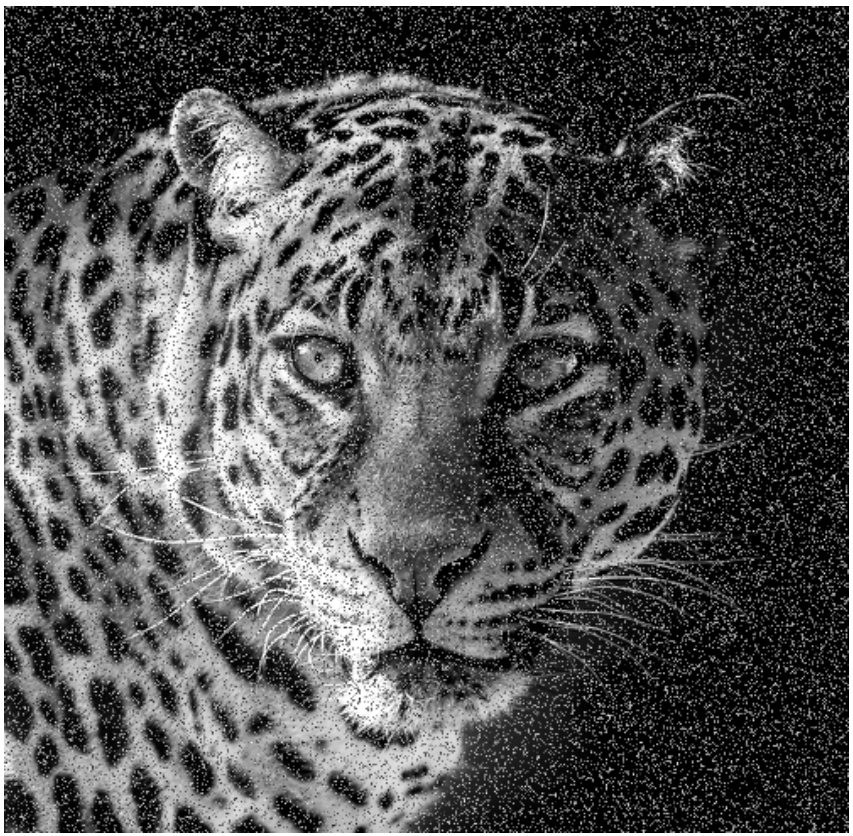
Denoised after 2 cycles



Noisy leopard

Denoising Leopard using $\hat{D}_p(f_p)$

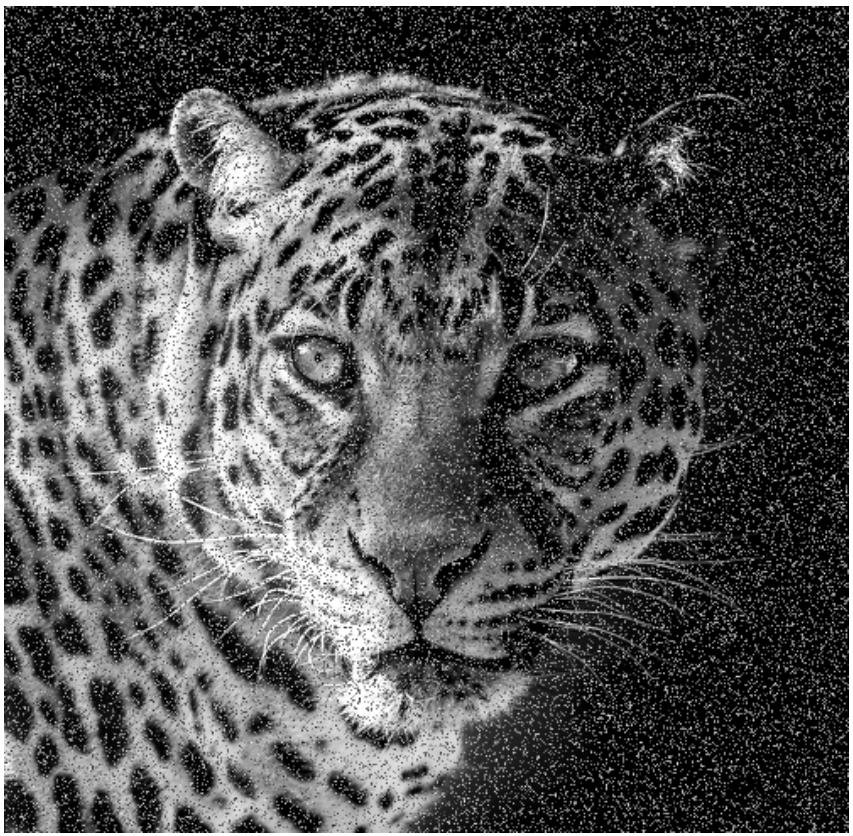
Denoised after 3 cycles



Noisy leopard

Denoising Leopard using $\hat{D}_p(f_p)$

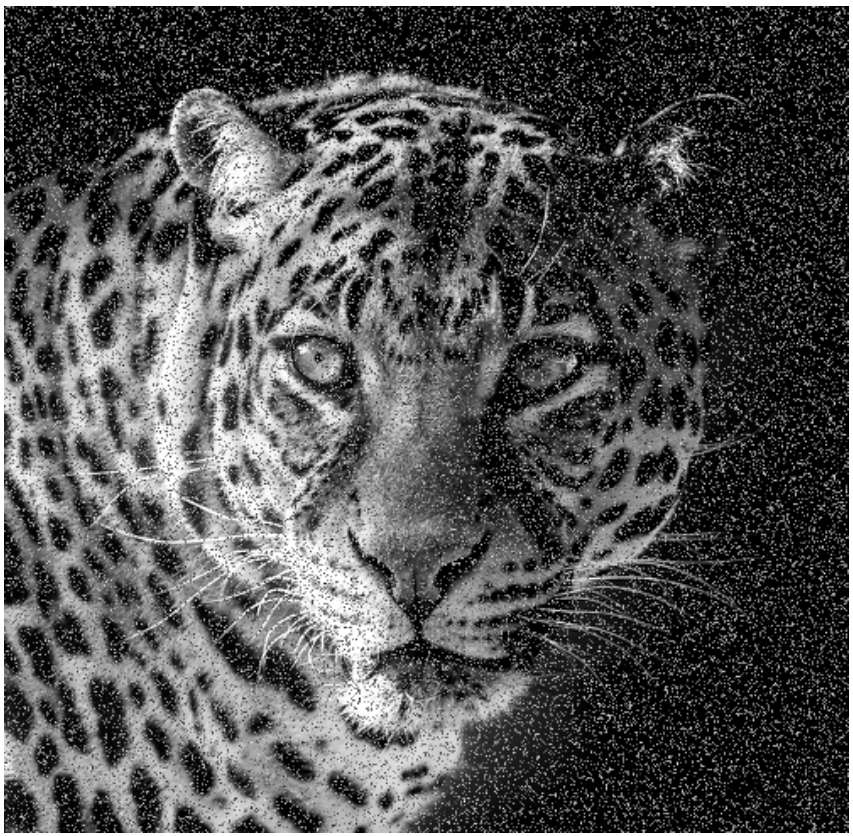
Denoised after 4 cycles



Noisy leopard

Denoising Leopard using $\hat{D}_p(f_p)$

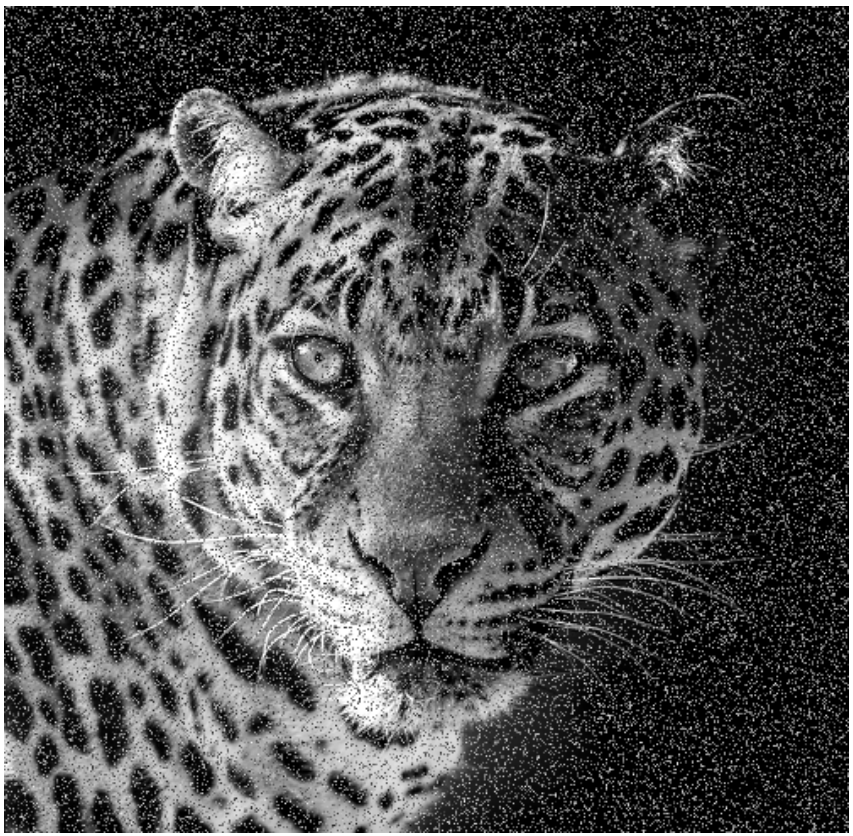
Denoised after 5 cycles



Noisy leopard

Denoising Leopard using $\hat{D}_p(f_p)$

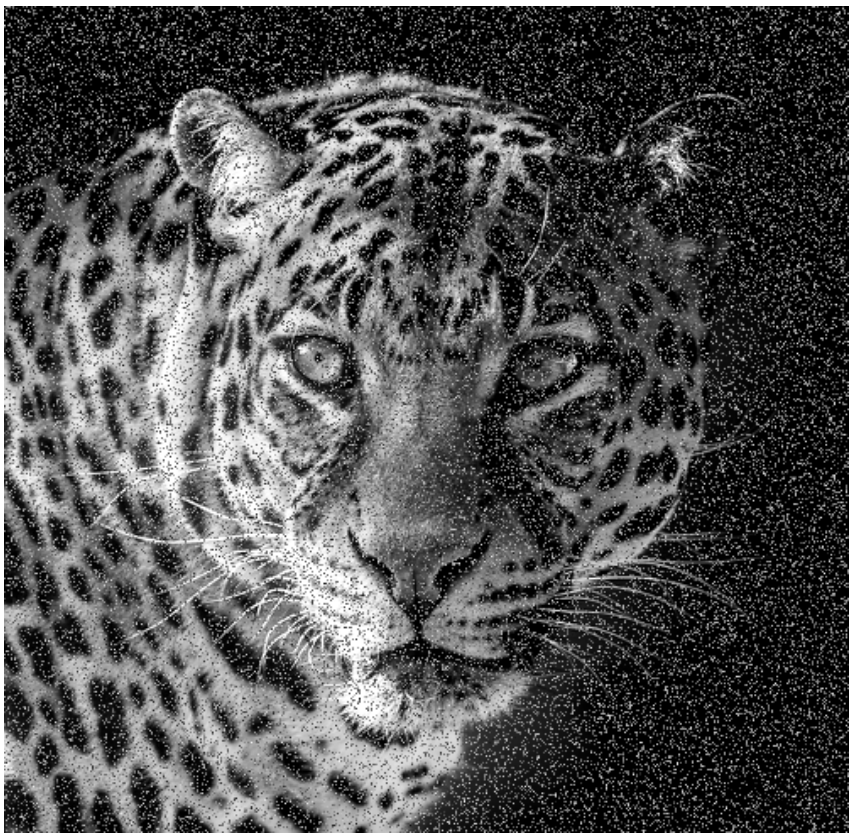
Denoised after 6 cycles



Noisy leopard

Denoising Leopard using $\hat{D}_p(f_p)$

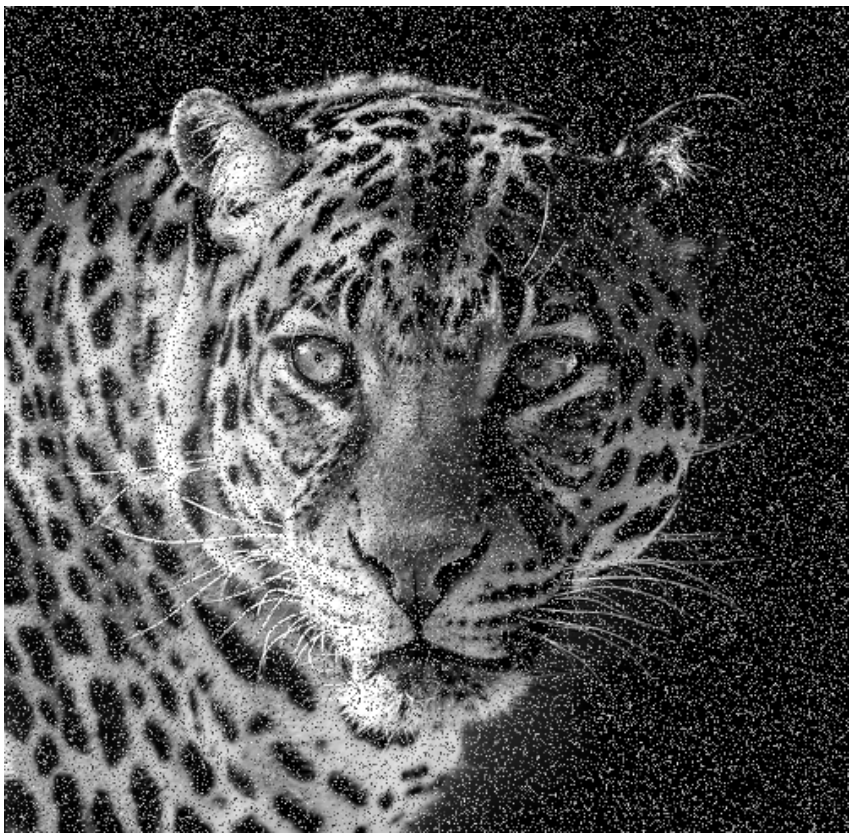
Denoised after 7 cycles



Noisy leopard

Denoising Leopard using $\hat{D}_p(f_p)$

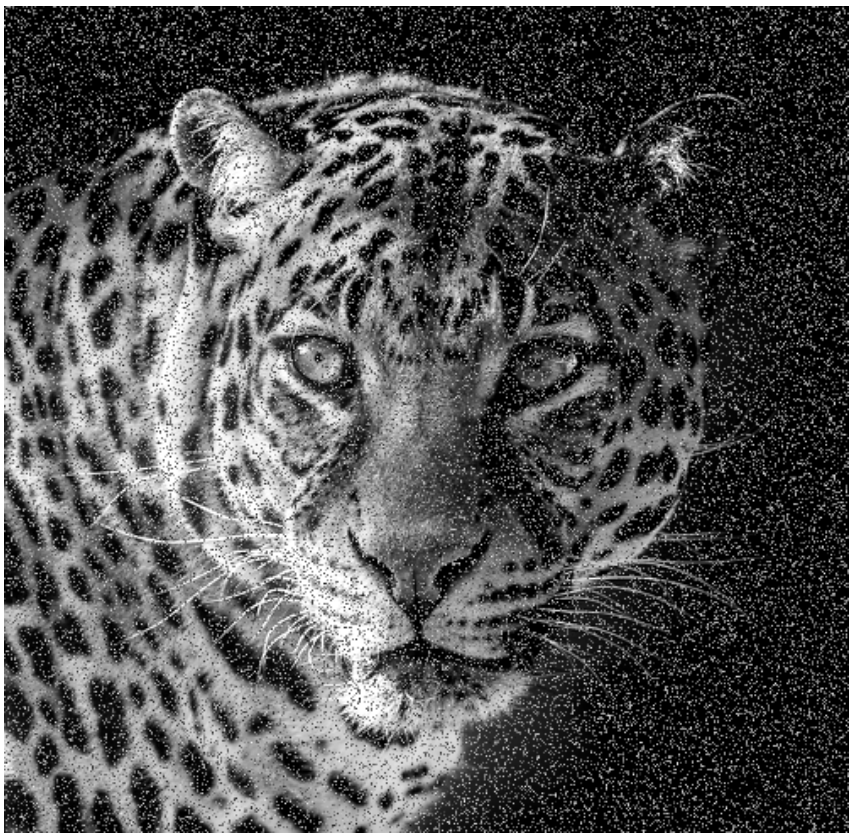
Denoised after 8 cycles



Noisy leopard

Denoising Leopard using $\hat{D}_p(f_p)$

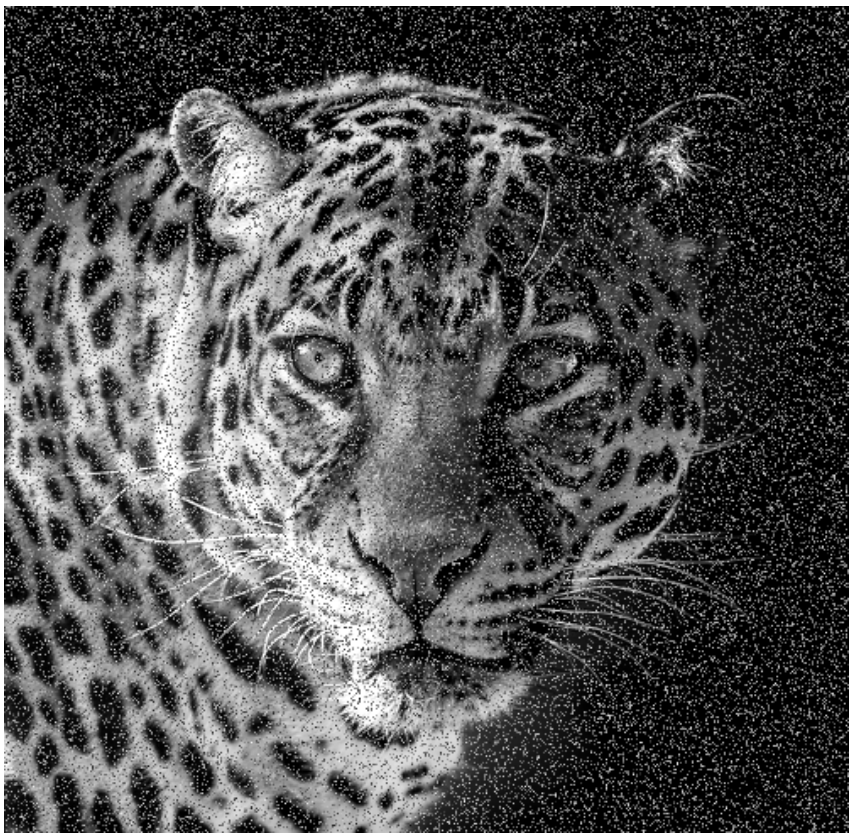
Denoised after 9 cycles



Noisy leopard

Denoising Leopard using $\hat{D}_p(f_p)$

Denoised after 10 cycles



Noisy leopard

Denoising Leopard using $\hat{D}_p(f_p)$

Denoised after 10 cycles

Orginal leopard

Denoising Lena $\hat{D}_p(f_p)$

Lena in 500x500 pixels

Denoised after 10 cycles

10% noisy Lenna

Denoising Lena $\hat{D}_p(f_p)$

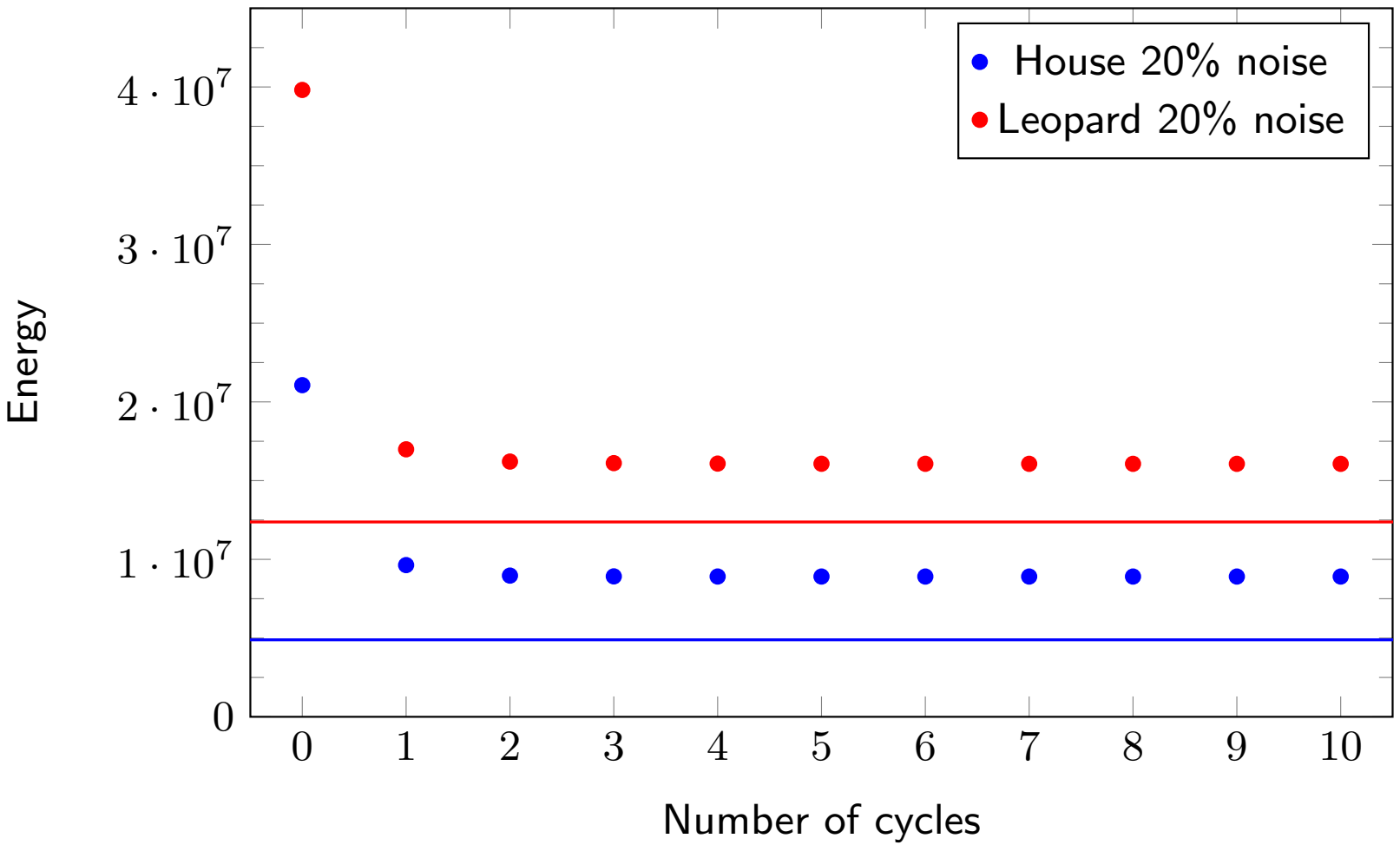
Lena in 500x500 pixels

Denoised after 10 cycles

Orginal Lenna

Energy over Cycles

Straight line is energy of original clear image



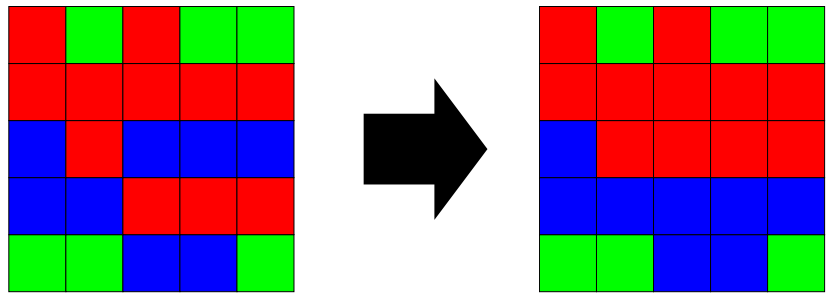
Conclusion

Conclusion

$$E(f) = E_{\text{smooth}}(f) + E_{\text{data}}(f)$$

Conclusion

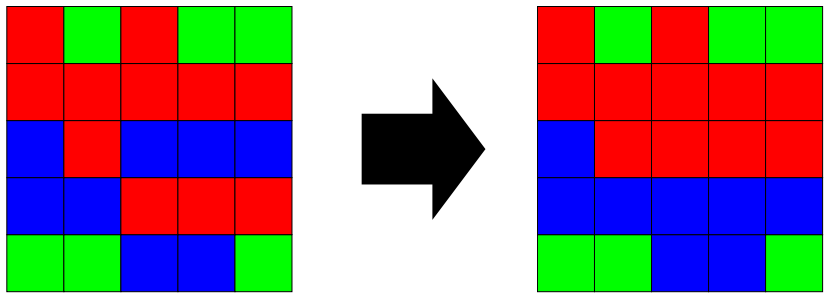
$$E(f) = E_{\text{smooth}}(f) + E_{\text{data}}(f)$$



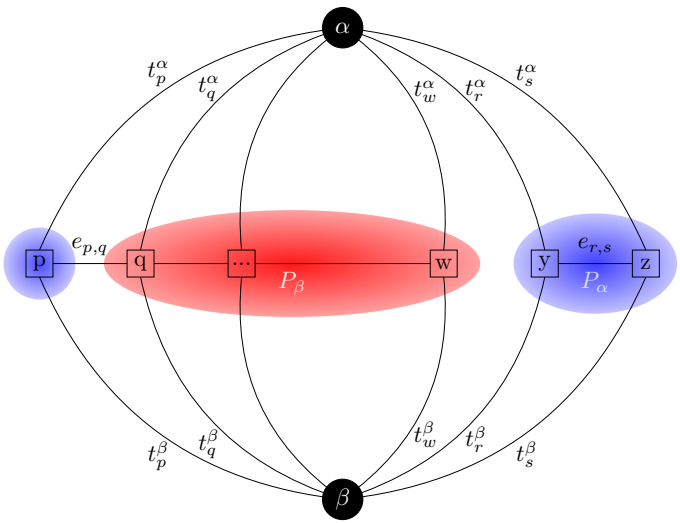
labeling f labeling $\hat{f}$

Conclusion

$$E(f) = E_{\text{smooth}}(f) + E_{\text{data}}(f)$$

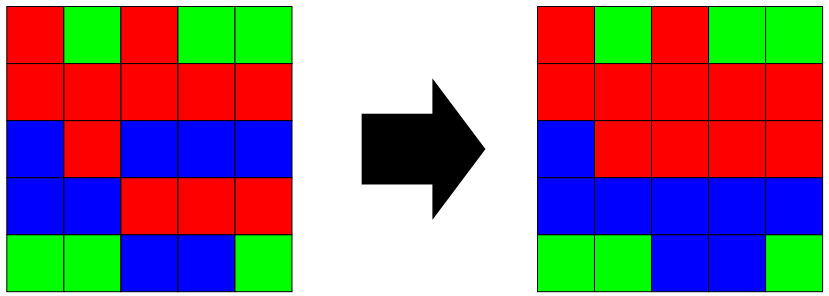


labeling f labeling $\hat{f}$

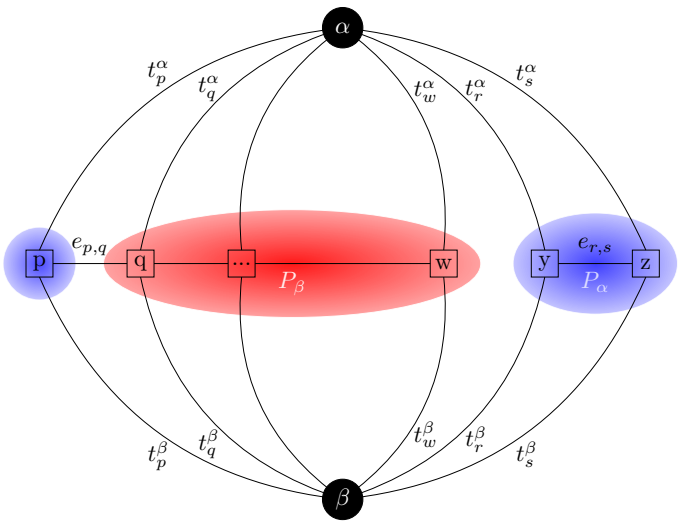


Conclusion

$$E(f) = E_{\text{smooth}}(f) + E_{\text{data}}(f)$$



labeling f labeling $\hat{f}$

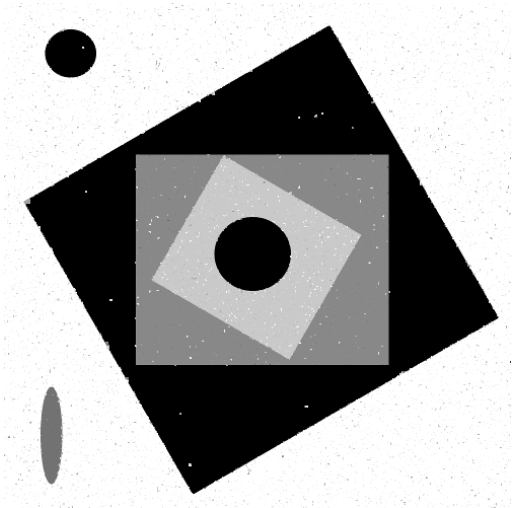


Bibliography

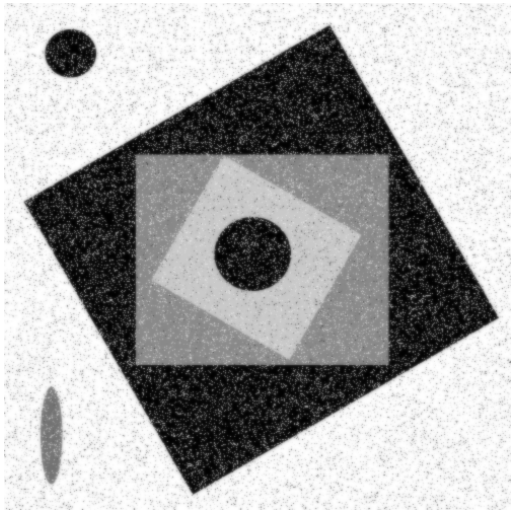
- Yuri Boykov and Vladimir Kolmogorov. "An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 26.9 (Sept. 2004), pp. 1124–1137. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2004.60. URL: http://dx.doi.org/10.1109/TPAMI.2004.60.
- Yuri Boykov, Olga Veksler, and Ramin Zabih. "Fast Approximate Energy Minimization via Graph Cuts". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 23.11 (2001), pp. 1222–1239. ISSN: 0162-8828.
- Emden R. Gansner and Stephen C. North. "An open graph visualization system and its applications to software engineering". In: *SOFTWARE - PRACTICE AND EXPERIENCE* 30.11 (2000), pp. 1203–1233. URL: www.graphviz.org.
- Vladimir Kolmogorov. "Graph Based Algorithms for Scene Reconstruction from Two or More Views". AAI3114475. PhD thesis. Ithaca, NY, USA, 2004.
- Vladimir Kolmogorov. *Maxflow*. Version 3.04. Oct. 30, 2017. URL: http://pub.ist.ac.at/~vnk/software.html.
- Olga Veksler. *Efficient Graph-Based Energy Minimization Methods In Computer Vision*. 1999.

Appendix

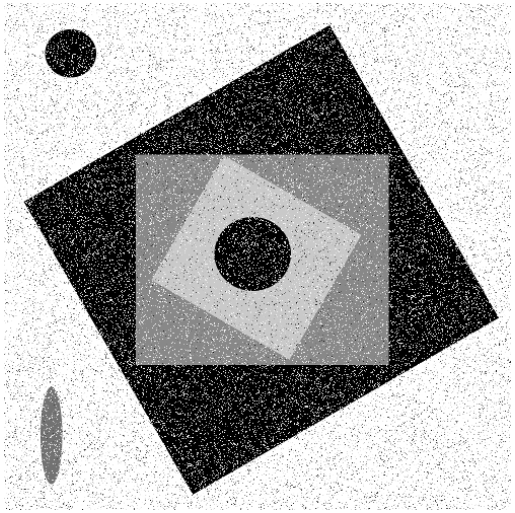
More examples



Denoised after 10 cycles using $\hat{D}_p(f_p)$

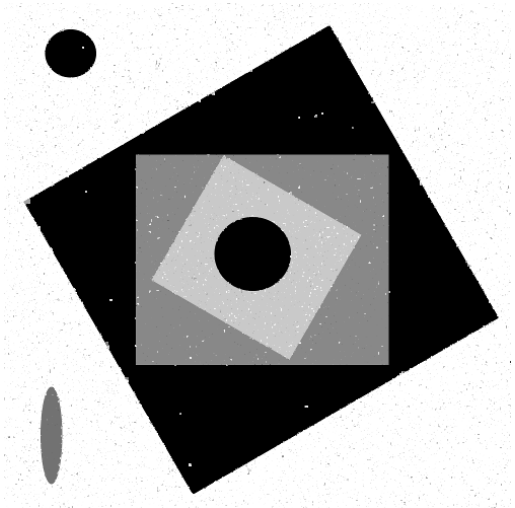


Denoised after 10 cycles using $D_p(f_p)$ and euclidean distance for V

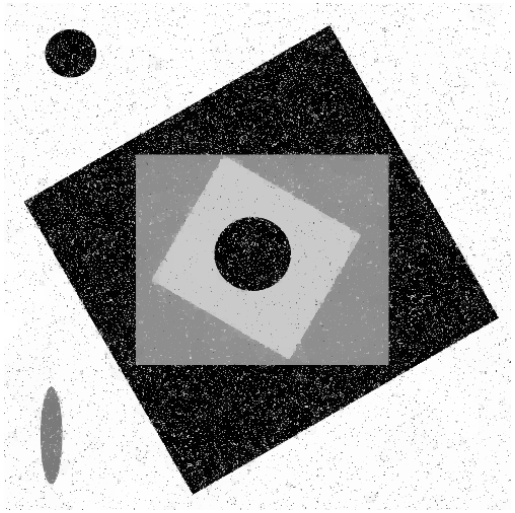


15% noised shapes

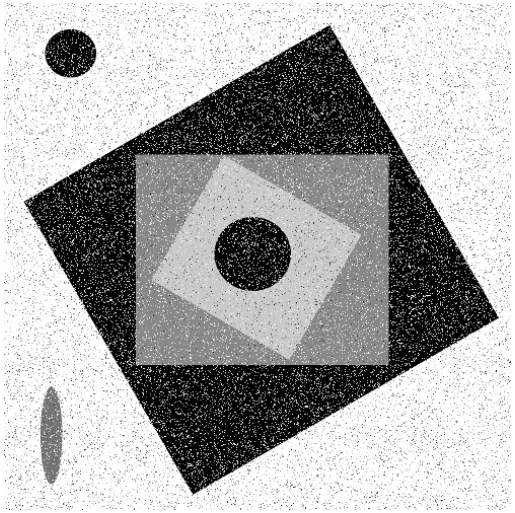
More examples



Denoised after 10 cycles using $\hat{D}_p(f_p)$

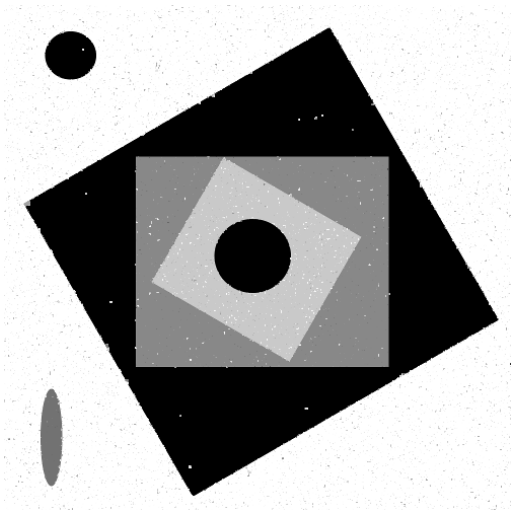


Denoised after 10 cycles using $D_p(f_p)$ and $c = 45$ in $V(f_p, f_q)$

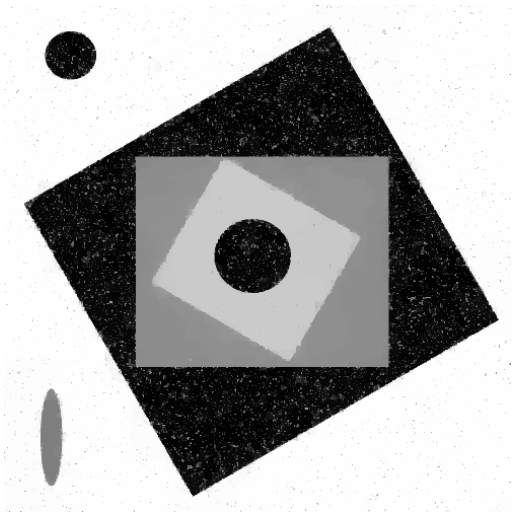


15% noised shapes

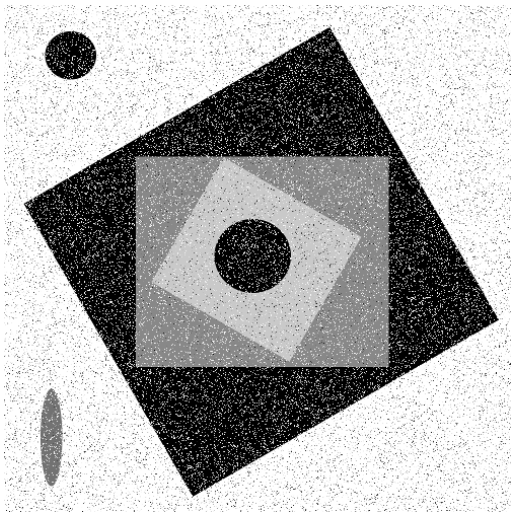
More examples



Denoised after 10 cycles using $\hat{D}_p(f_p)$

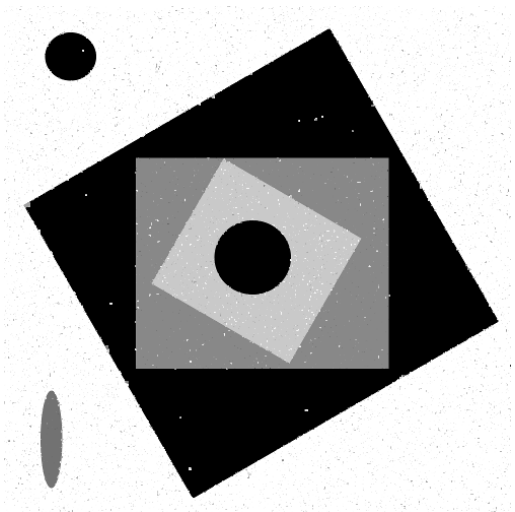


Denoised after 10 cycles using $D_p(f_p)$ and $c = 80$ in $V(f_p, f_q)$

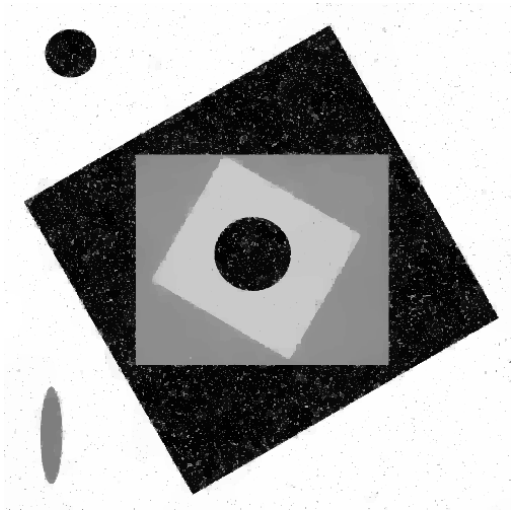


15% noised shapes

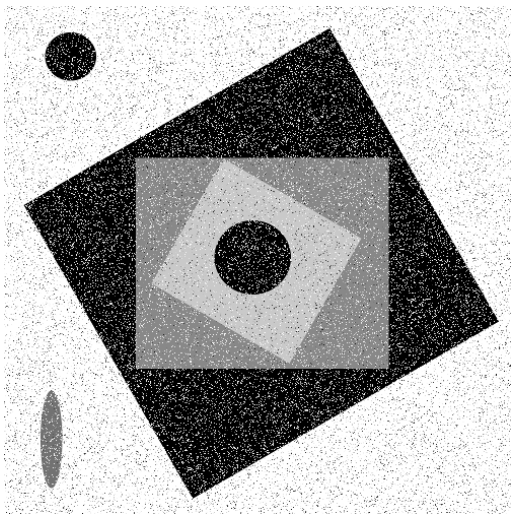
More examples



Denoised after 10 cycles using $\hat{D}_p(f_p)$



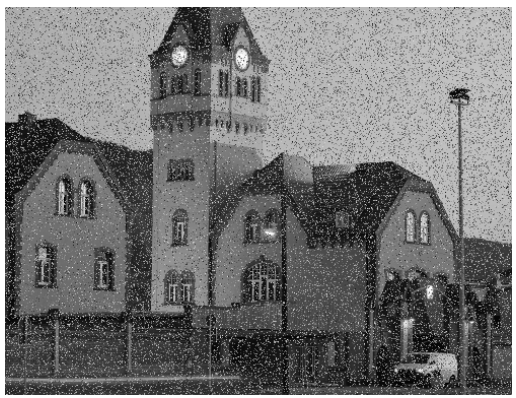
Denoised after 10 cycles using $D_p(f_p)$ and min 500 euclidean distance for $V(f_p, f_q)$



15% noised shapes

Example House

Denoised after 10 cycles using $\hat{D}_p(f_p)$

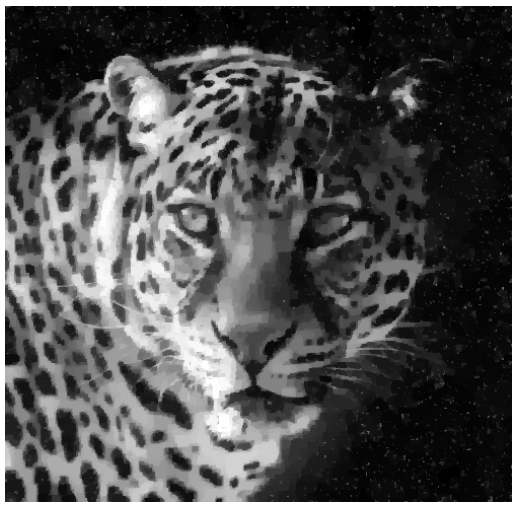


20% noised house

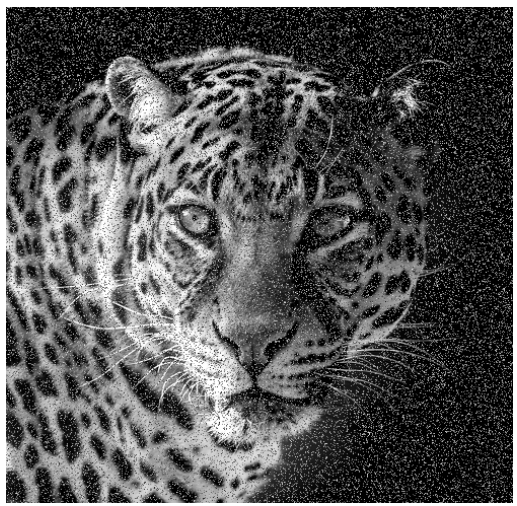
Original house

Comparison House

Denoised after 10 cycles using $\hat{D}_p(f_p)$



Denoised after 10 cycles using $\hat{D}_p(f_p)$ and $V = 80 \cdot |f_p - f_q|$



Original image