Image Collections and Texture Synthesis

TI3135TU Visual Data Processing

Lecture 4

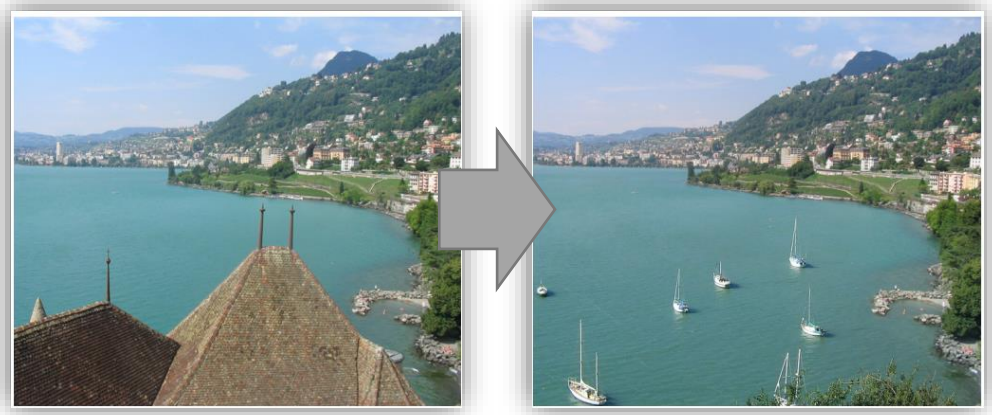
2024/2025

Petr Kellnhofer

Lecture plan

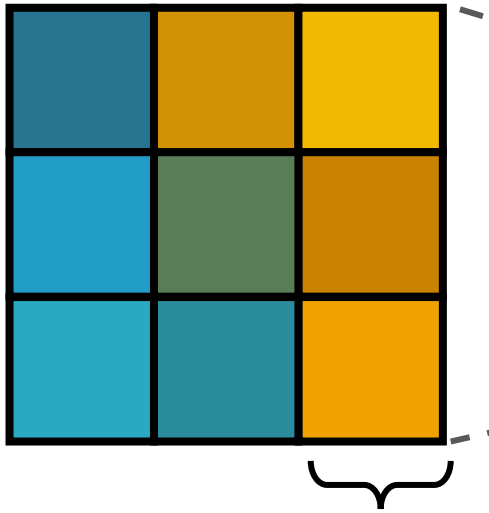
- **Instructor:**
  - Petr Kellnhofer (p.kellnhofer@tudelft.nl)
    - Assistant Professor in the Computer Graphics and Visualization Group

- **Content**
  - What is image inpainting?
  - How to do inpainting:
    - Using a single image.
    - Using image collection.
    - Using a neural network.



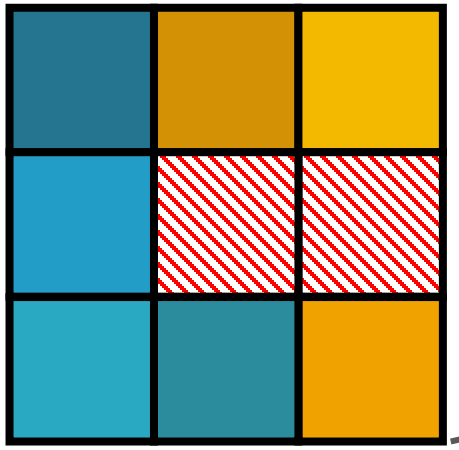
Refresher: Image representation

- How are images stored in computer memory?



Refresher: Image representation

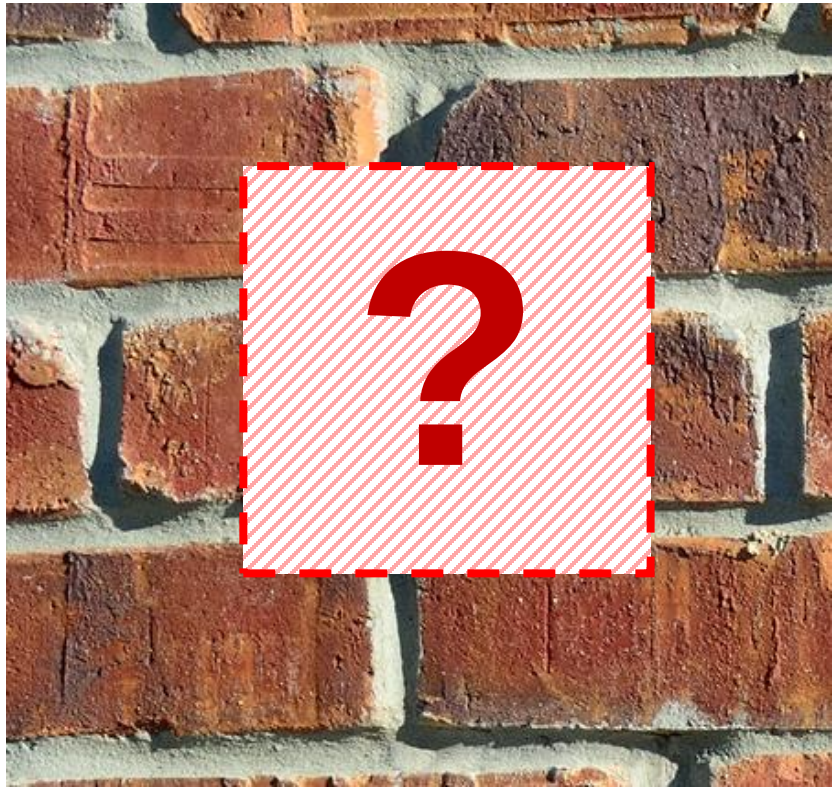
- How are images stored in computer memory?



- What if some pixels go missing?

Can you complete an image?

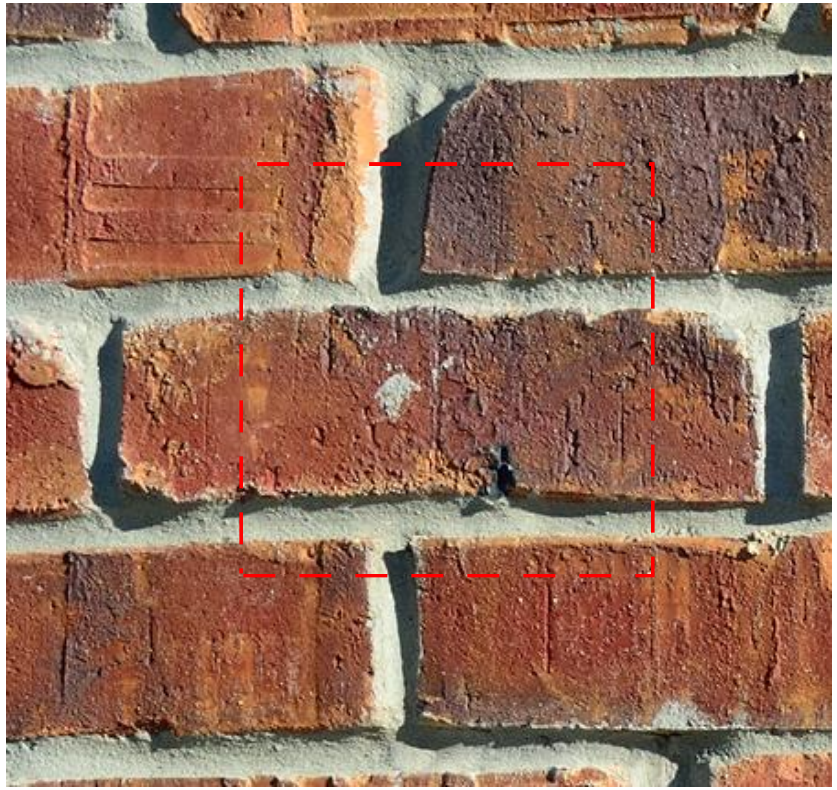
- Which of the following patches fit the best and **why**?



A

B

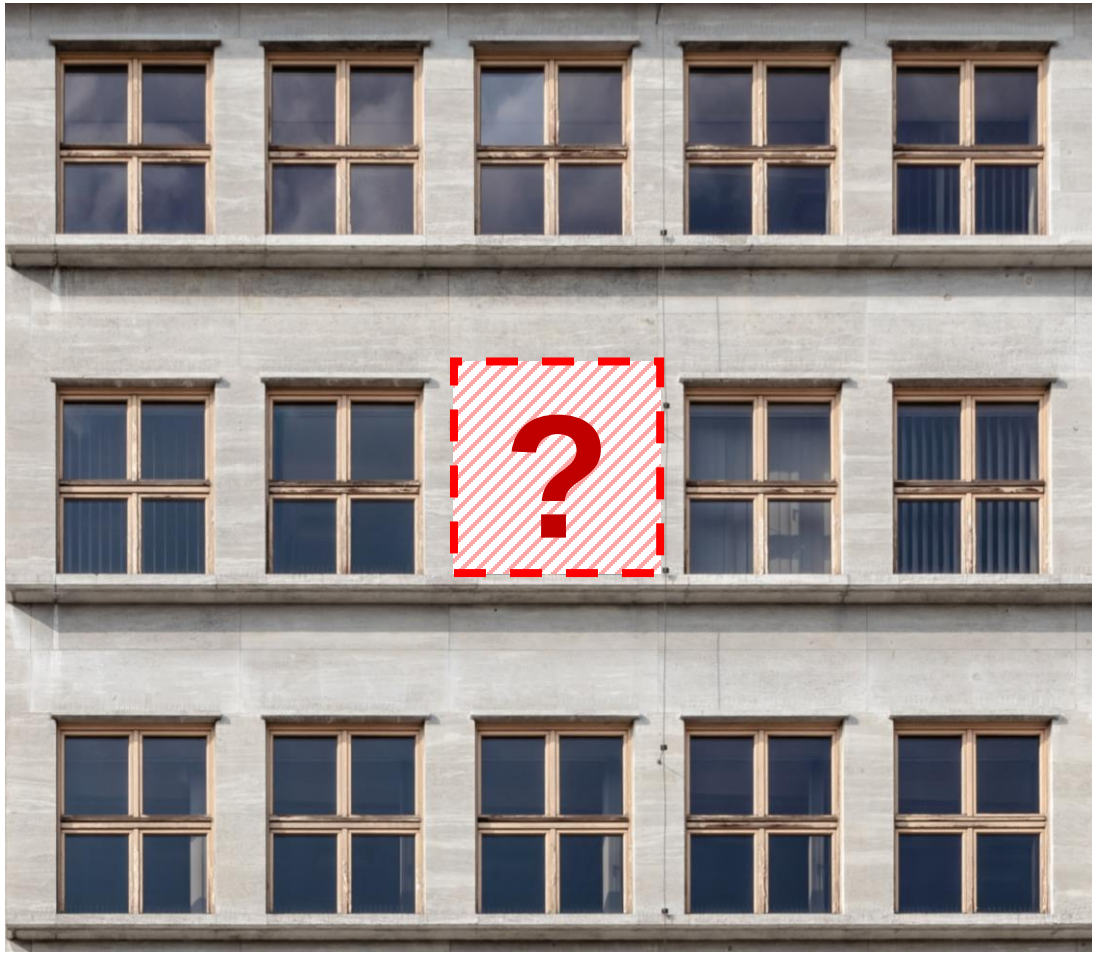
C



A

B

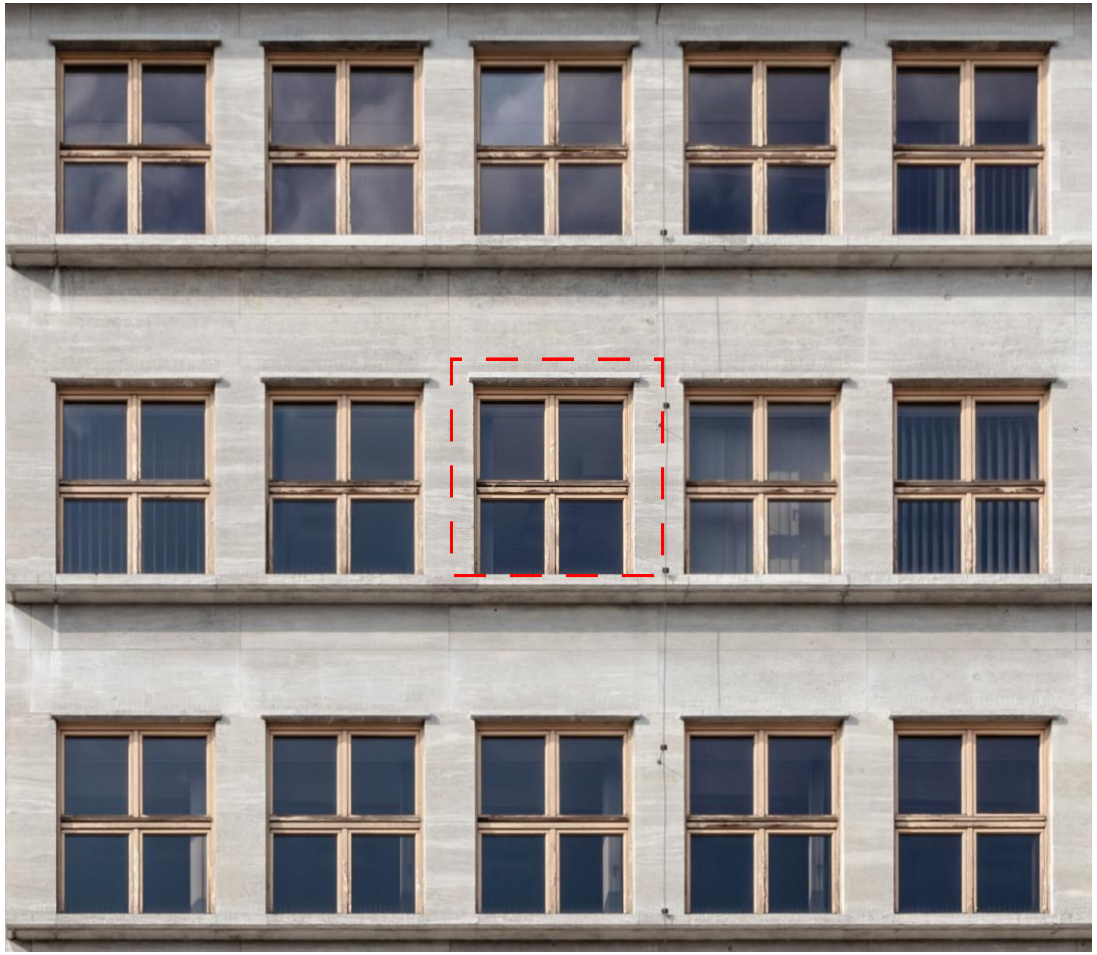
C



A

B

C



A

B

C

A

B

C

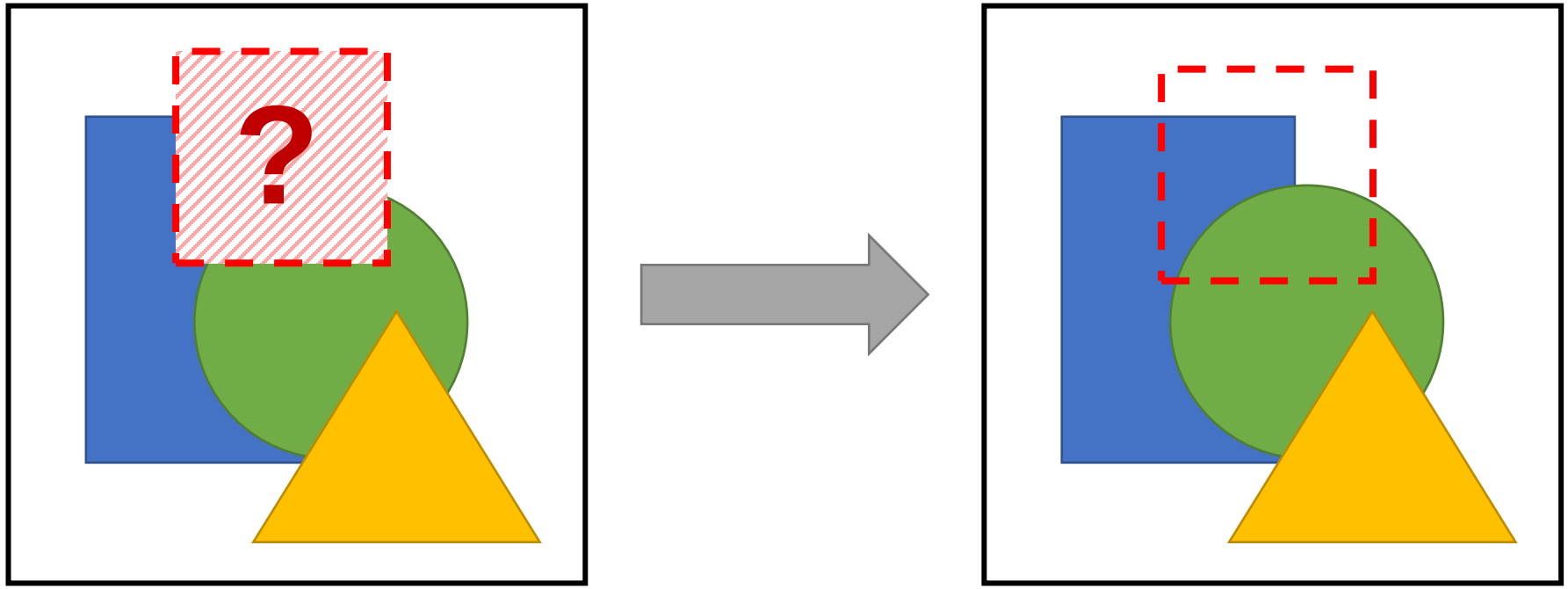
A

B

C

Image inpainting

... is a process of completing missing image parts.



Application: Image Restoration

Credit: Bruce C. Cooper @ Wikipedia.org

Credit: Michel Vuijlsteke @ Wikipedia.org

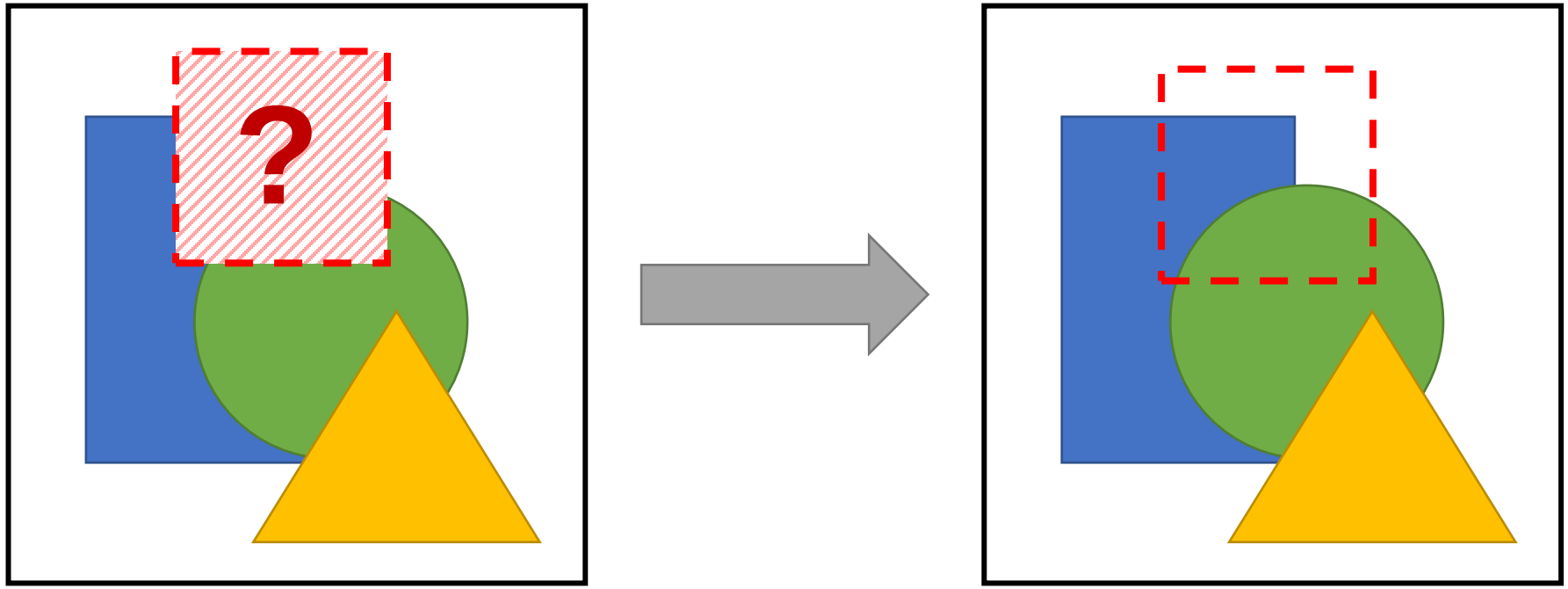
Application: Image editing

Fine Art Images/Heritage Images/Getty Images & AFP/GettyImages

James Hays and Alexei A. Efros: Scene Completion Using Millions of Photographs. SIGGRAPH 2007.

Image inpainting

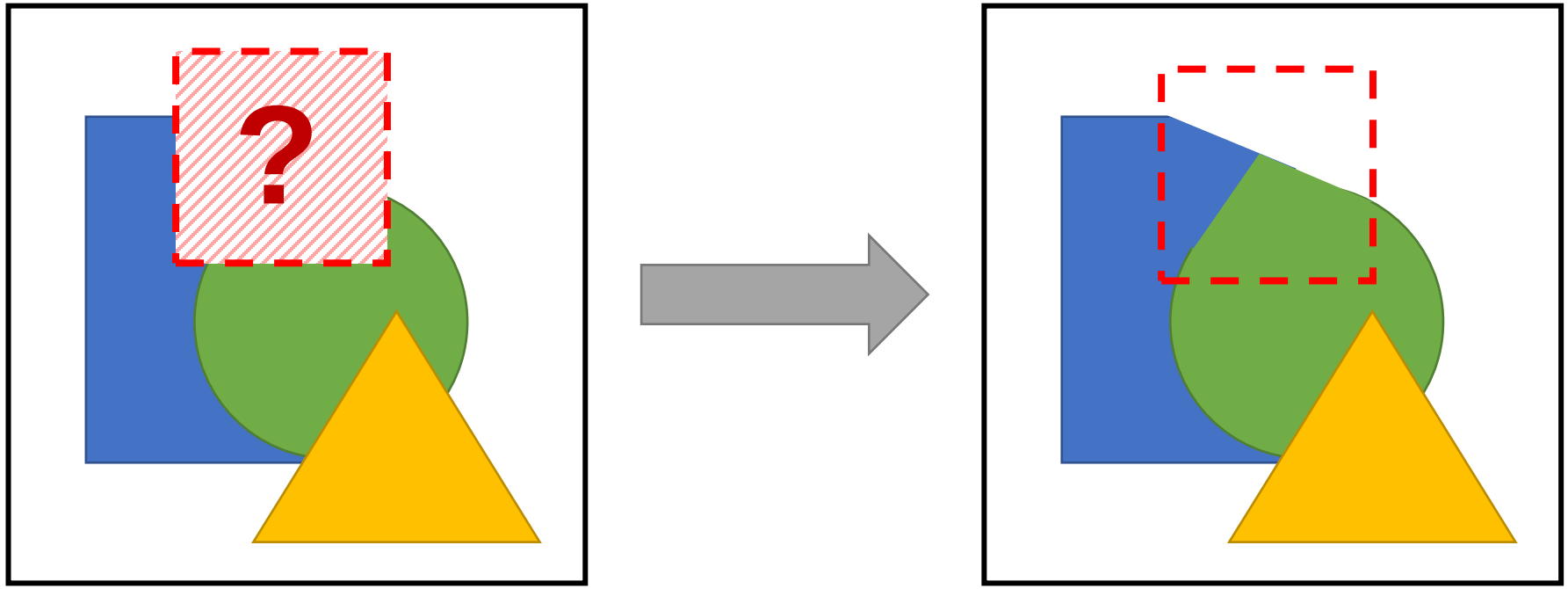
... is a process of completing missing image parts.



A valid solution.

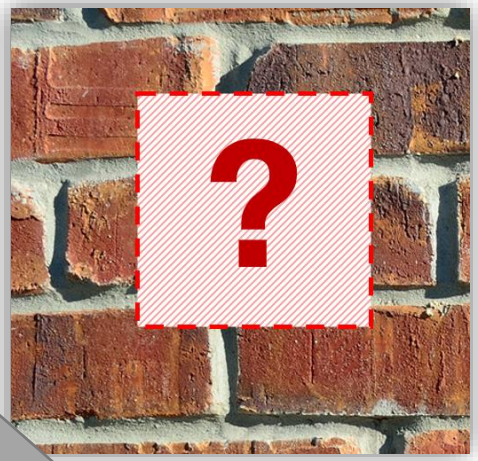
Image inpainting

... is a process of completing missing image parts.



Also a valid solution.

Inpainting cues



"The edges/gradients should match."

Simple statistics (mean, variance).

"The pattern should match."

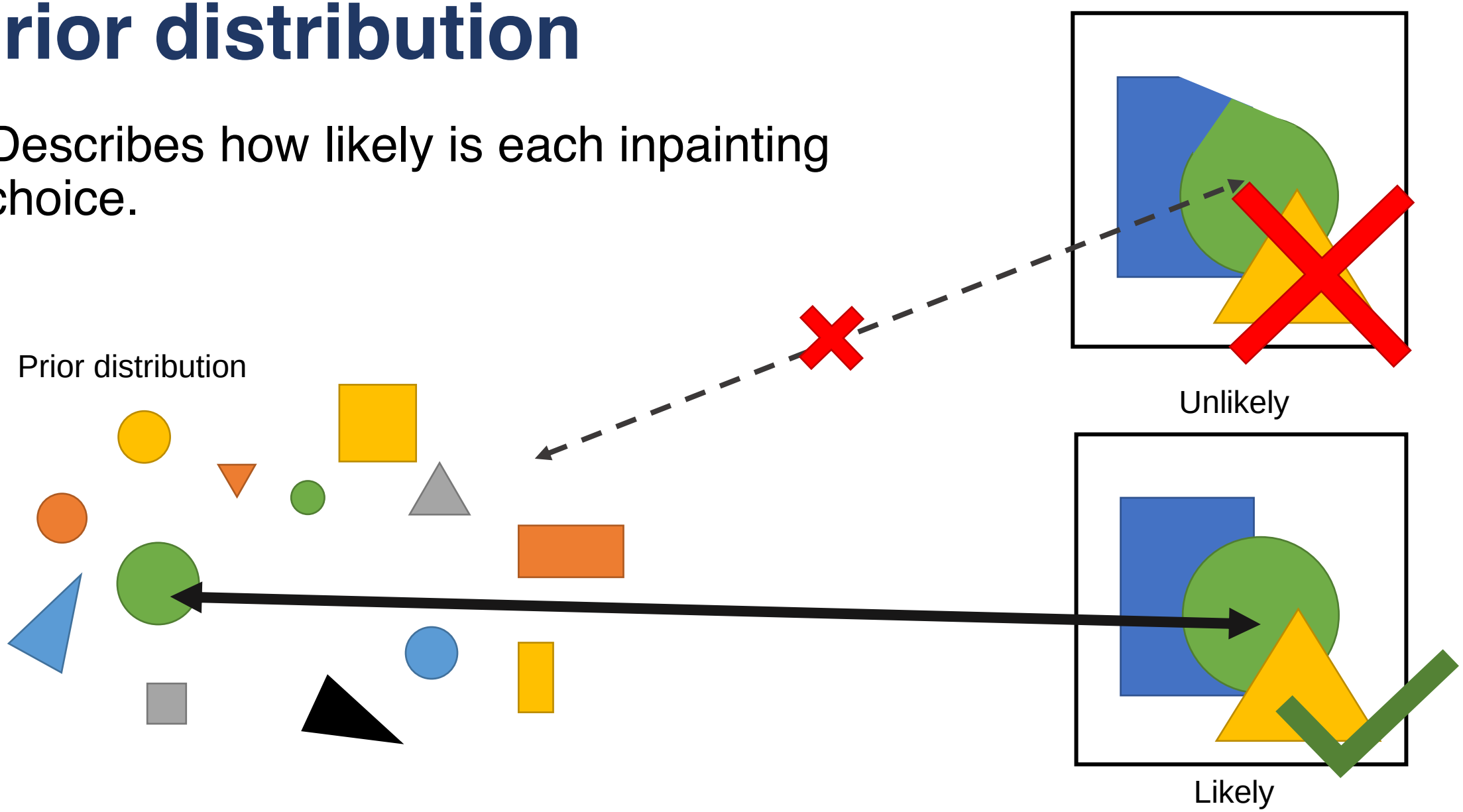
More complex statistics (patterns,…).

"The semantic should match."

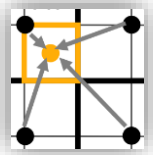
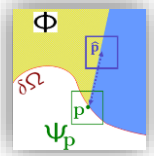
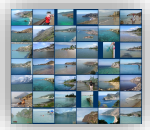
High-level statistics.

Prior distribution

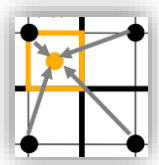
- Describes how likely is each inpainting choice.



Algorithms

	Statistic		Prior
1. Pull-Push	Simple: Mean color		Single image
2. PDE inpainting	Simple: Gradients		Single image
3. Exemplar inpainting	Medium: Patterns		Single image
4. Collection sampling	Medium: Patterns		Collection
5. Neural inpainting	Complex: Multi-scale		Collection

Algorithms

	Statistic		Prior
1. Pull-Push	Simple: Mean color		Single image
2. PDE inpainting	Simple: Gradients		Single image
3. Exemplar inpainting	Medium: Patterns		Single image
4. Collection sampling	Medium: Patterns		Collection
5. Neural inpainting	Complex: Multi-scale		Collection

Pull-push algorithm

Idea: Match the mean color of the surrounding.

Free-Photos @ Pixabay

0.9	0.9	0.8	0.9
0.9	0.8	0.8	0.8
0.1	0.2	0.2	0.3
0.1	0.2	0.1	0.3

Gortler, Steven J., et al. "The lumigraph." Proceedings of the 23rd annual conference on Computer graphics and interactive techniques. 1996.

Pull-push algorithm

Idea: Match the mean color of the surrounding.

Free-Photos @ Pixabay

0.9	0.9		
0.9			
0.1	0.2	0.2	0.3
0.1	0.2	0.1	0.3

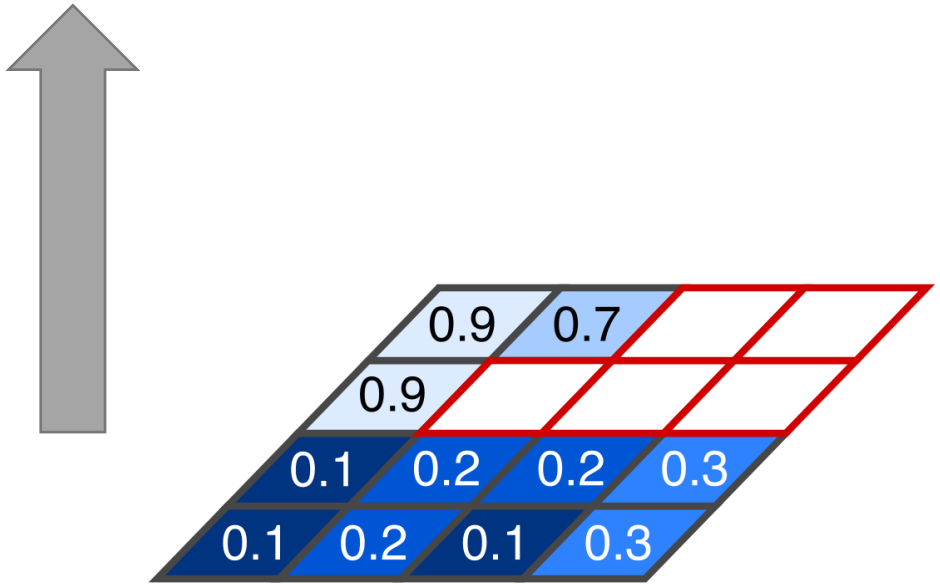
Gortler, Steven J., et al. "The lumigraph." Proceedings of the 23rd annual conference on Computer graphics and interactive techniques. 1996.

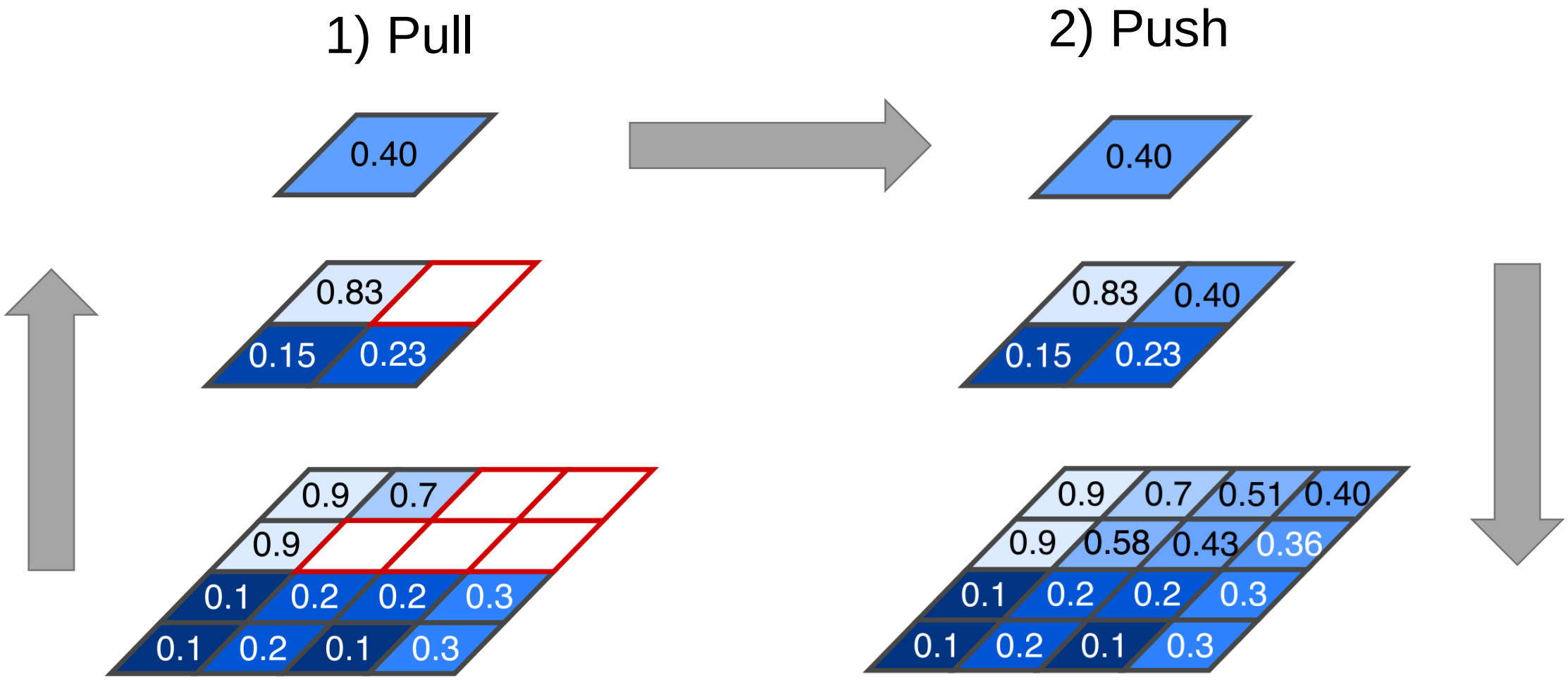
Pull-push algorithm

Idea: Match the mean color of the surrounding.

0.9	0.9		
0.9			
0.1	0.2	0.2	0.3
0.1	0.2	0.1	0.3

Pull-push algorithm



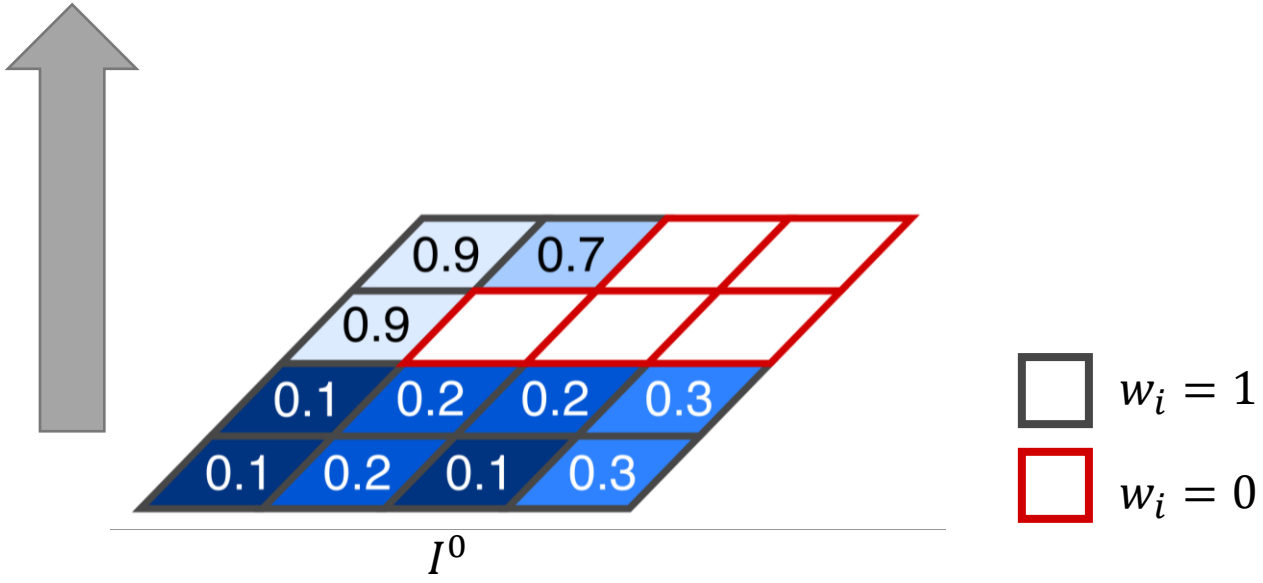


Aggregate information.

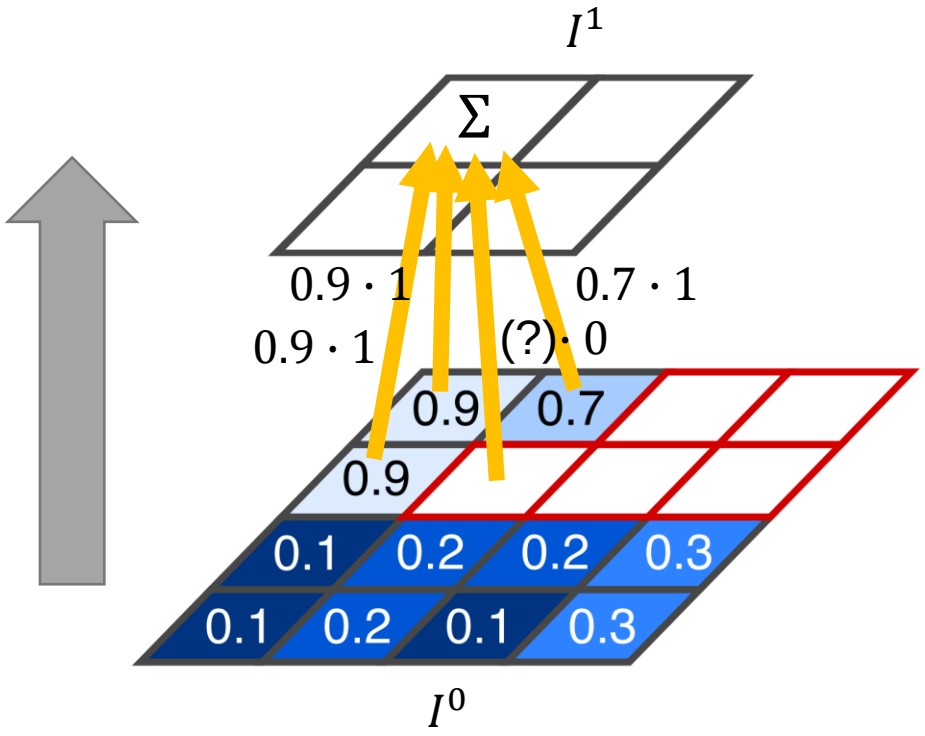
Distribute information.

1) Pull

$$I^0 = I$$



1) Pull



$$I^0 = I$$

$$I^{k+1} = \frac{\sum_{i=0}^{4} I_i^k w_i^k}{\sum_{i=0}^{4} w_i^k} = \frac{0.9 \cdot 1 + 0.9 \cdot 1 + 0.7 \cdot 1 + (?) \cdot 0}{1 + 1 + 1 + 0} = 0.8\overline{3}$$

$$w^{k+1} = \min\left(\sum_{i=0}^{4} w_i^k, 1\right) = \min(1 + 1 + 1 + 0, 1) = 1$$

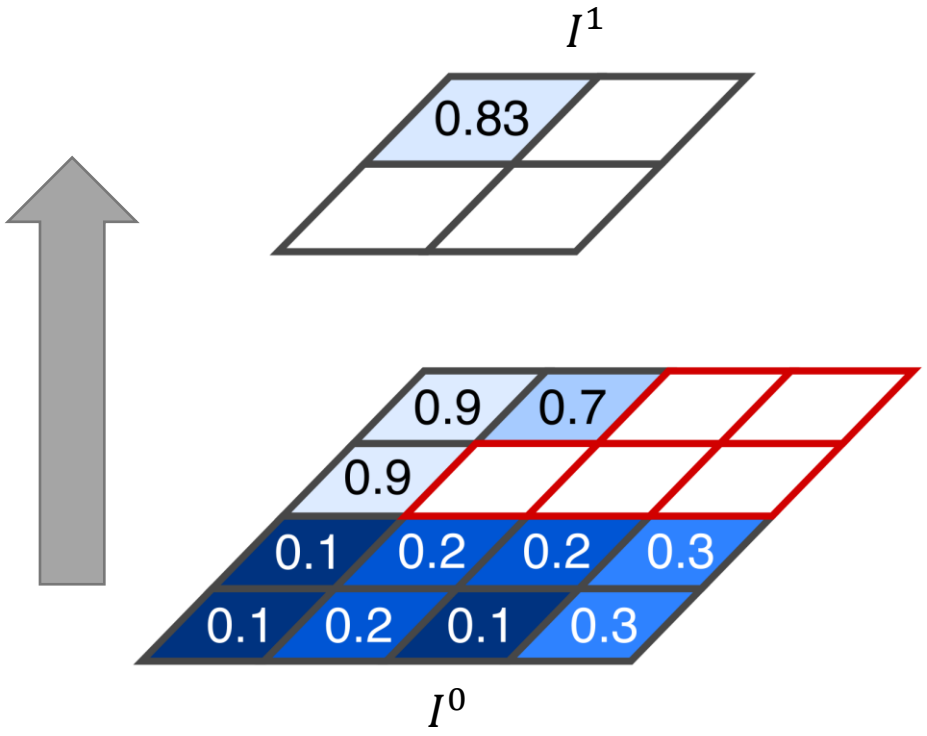
$w_i = 1$

$w_i = 0$

1) Pull

$$I^{k+1} = \frac{\sum_{i=0}^{4} I_i^k w_i^k}{\sum_{i=0}^{4} w_i^k} = \frac{0.9 \cdot 1 + 0.9 \cdot 1 + 0.7 \cdot 1 + (?) \cdot 0}{1 + 1 + 1 + 0} = 0.8\bar{3}$$

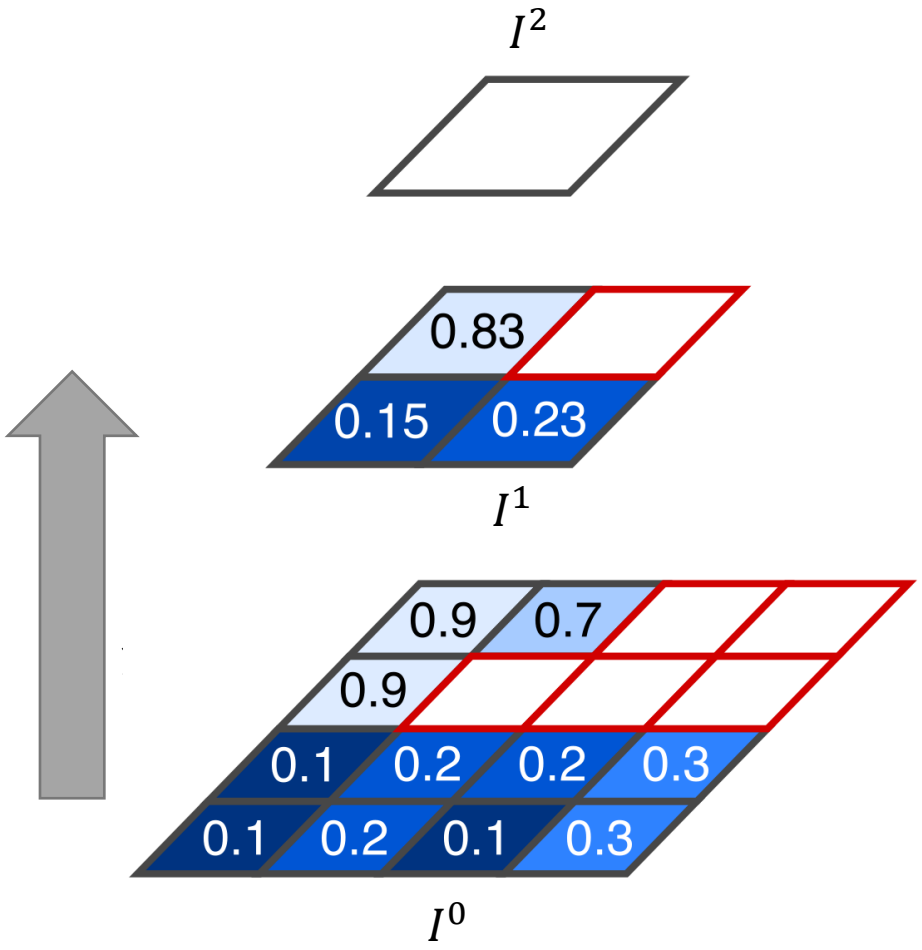
$$w^{k+1} = \min\left(\sum_{i=0}^{4} w_i^k, 1\right) = \min(1 + 1 + 1 + 0, 1) = 1$$



$w_i = 1$

$w_i = 0$

1) Pull



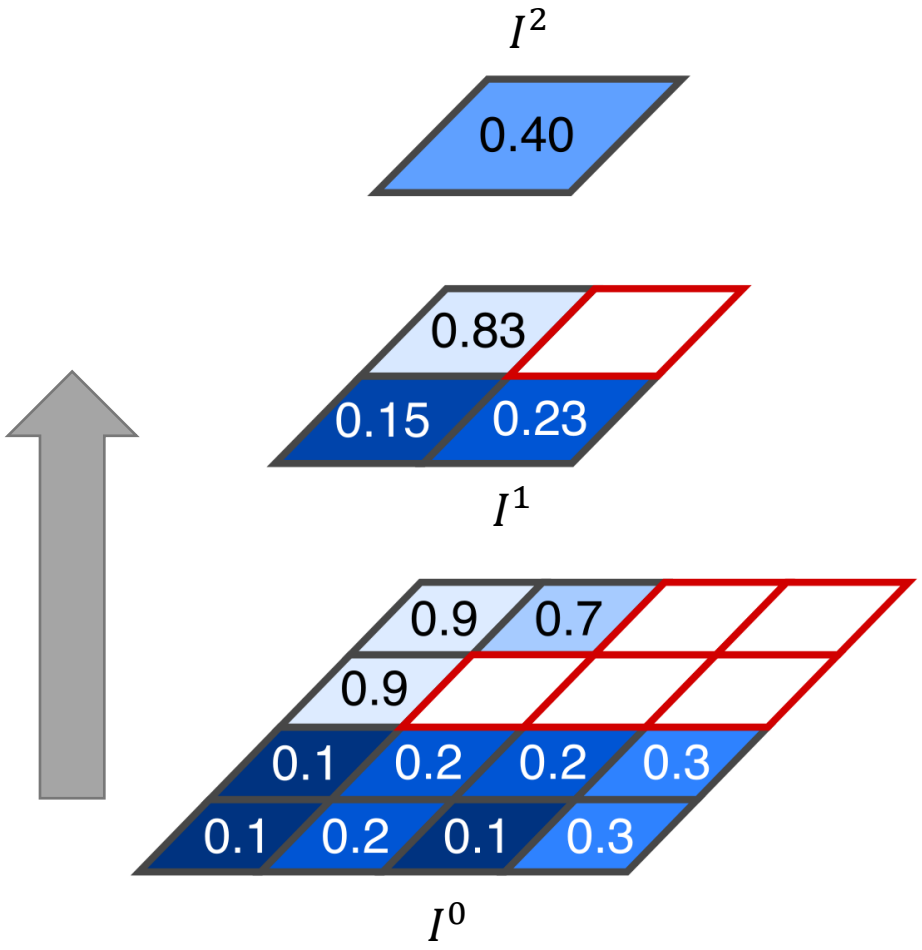
$$I^{k+1} = \frac{\sum_{i=0}^{4} I_i^k w_i^k}{\sum_{i=0}^{4} w_i^k} = \frac{0.83 \cdot 1 + 0.15 \cdot 1 + 0.23 \cdot 1 + (?) \cdot 0}{1 + 1 + 1 + 0} = 0.40\overline{3}$$

$$w^{k+1} = \min\left(\sum_{i=0}^{4} w_i^k, 1\right) = \min(1 + 1 + 1 + 0, 1) = 1$$

$w_i = 1$

$w_i = 0$

1) Pull



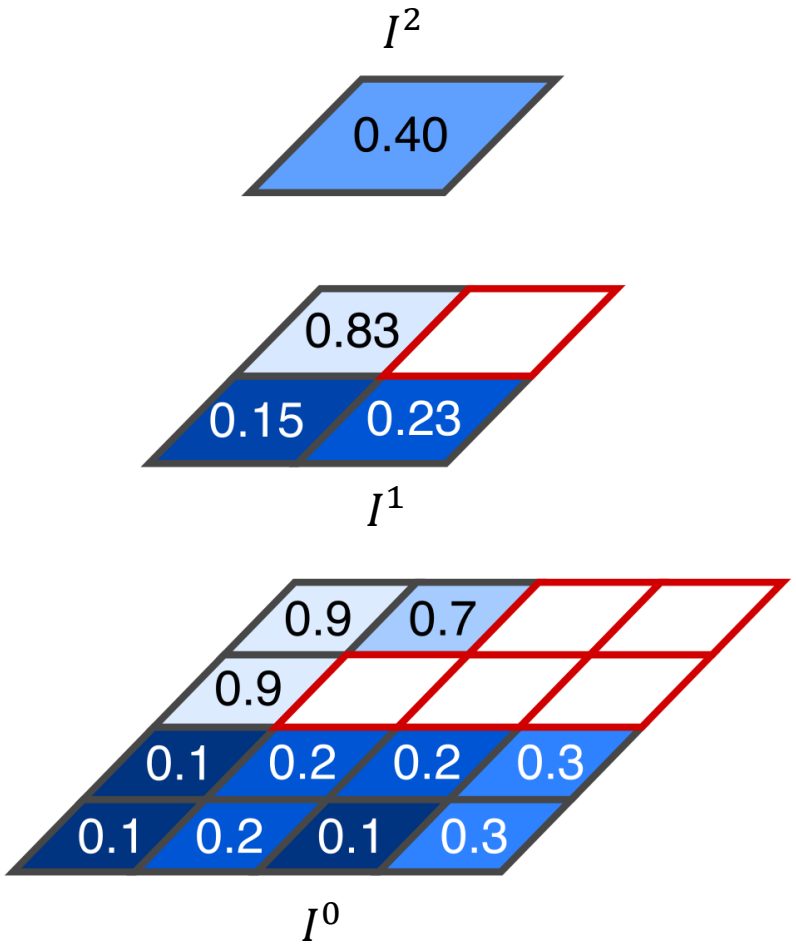
$$I^{k+1} = \frac{\sum_{i=0}^{4} I_i^k w_i^k}{\sum_{i=0}^{4} w_i^k} = \frac{0.83 \cdot 1 + 0.15 \cdot 1 + 0.23 \cdot 1 + (?) \cdot 0}{1 + 1 + 1 + 0} = 0.40\overline{3}$$

$$w^{k+1} = \min\left(\sum_{i=0}^{4} w_i^k, 1\right) = \min(1 + 1 + 1 + 0, 1) = 1$$

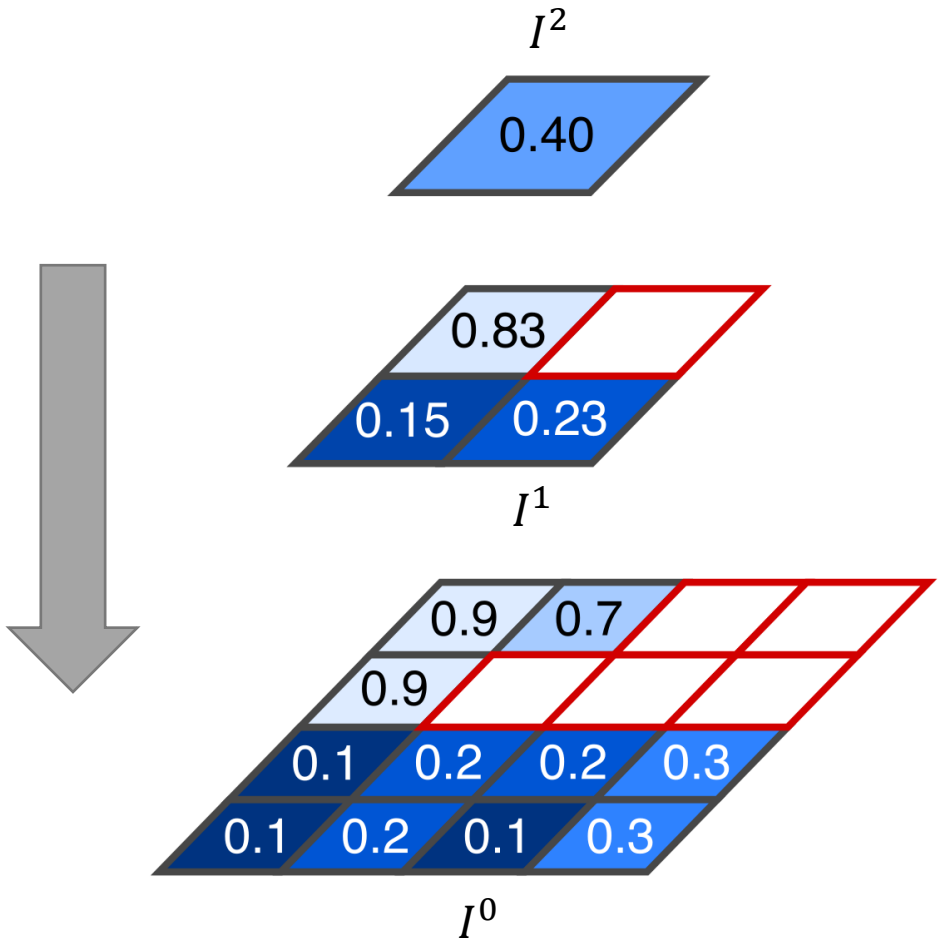
$w_i = 1$

$w_i = 0$

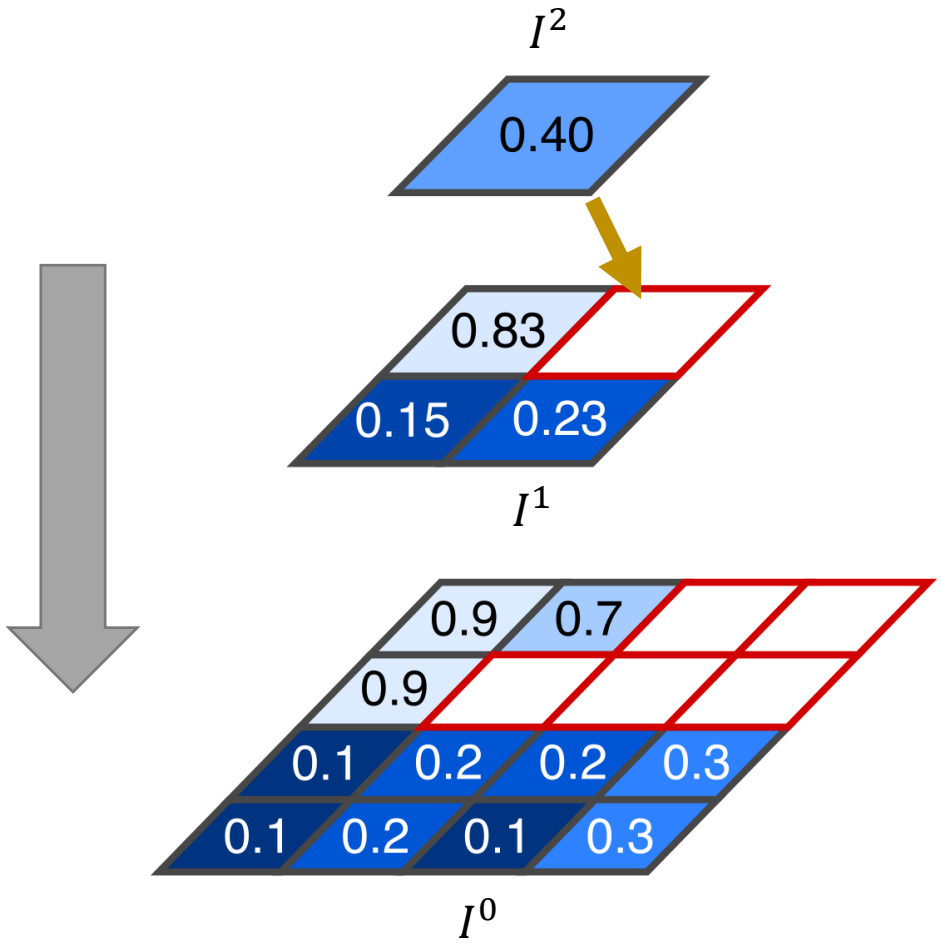
1) Pull



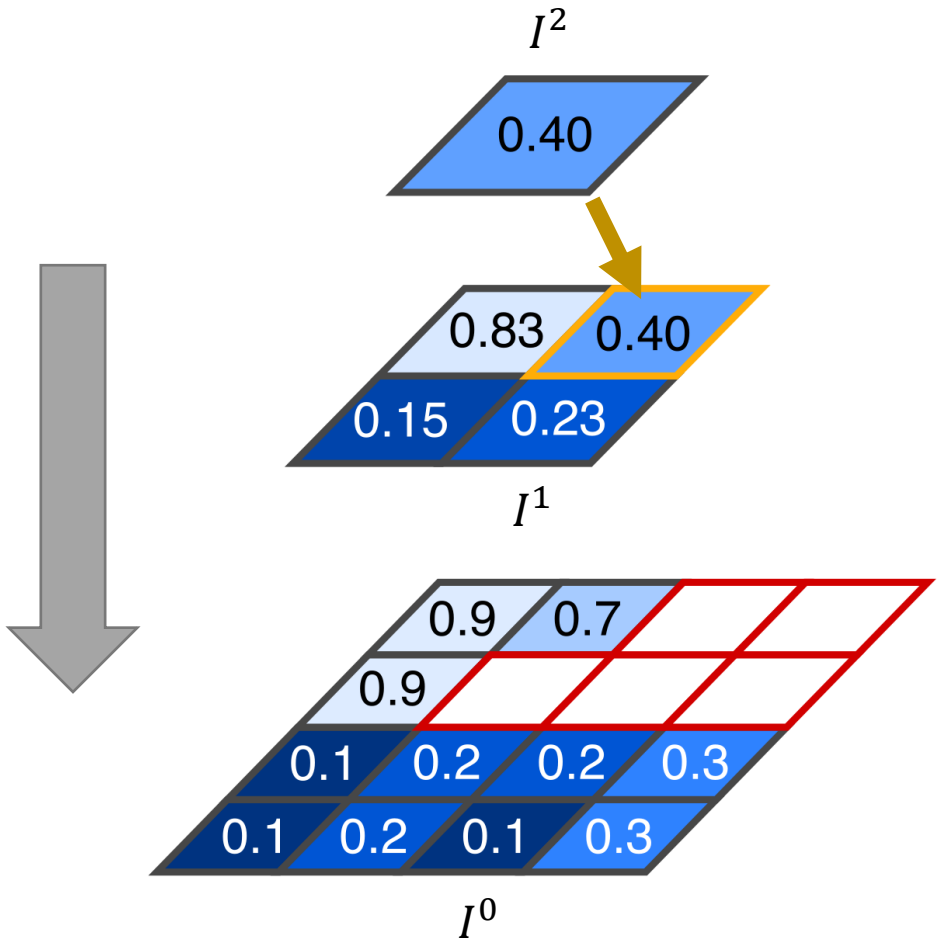
2) Push



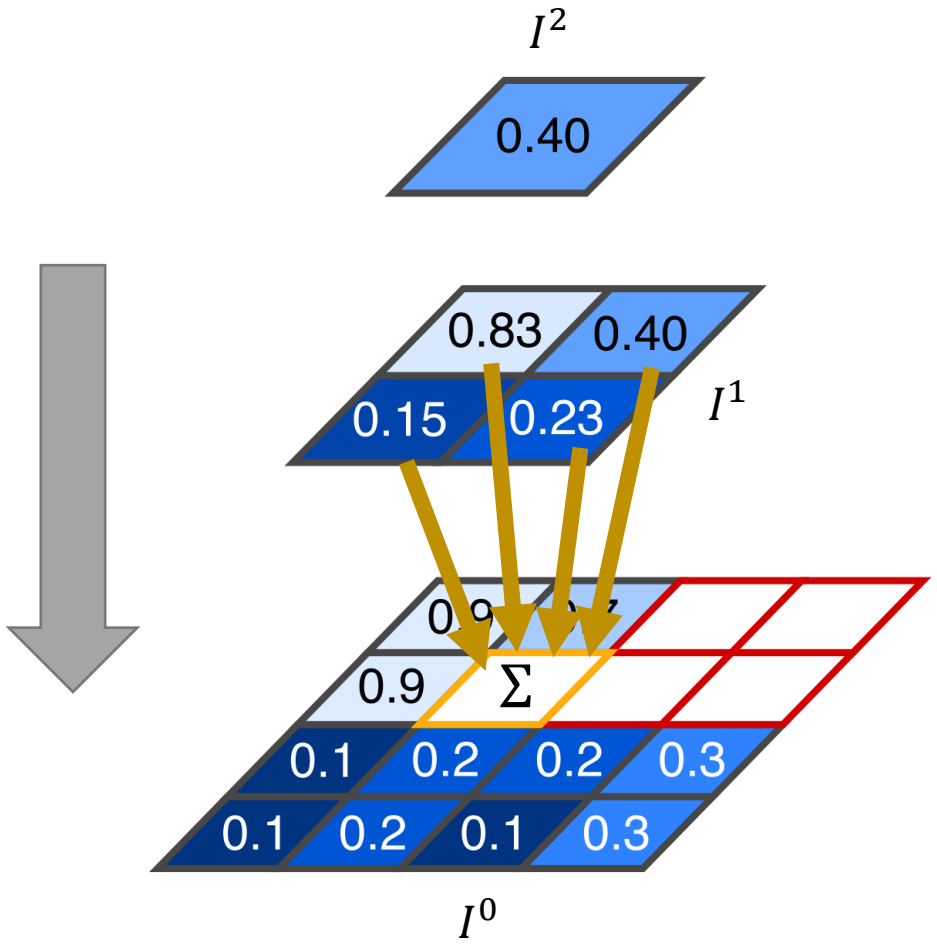
2) Push



2) Push



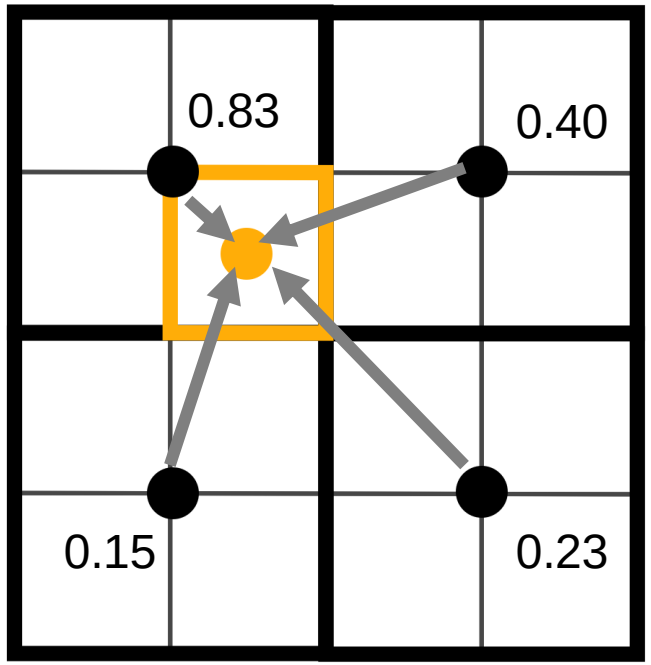
2) Push



Interpolate 4 samples from level $k+1$.

$$I^k = \sum_{i=0}^{4} I_i^{k+1} u_i$$

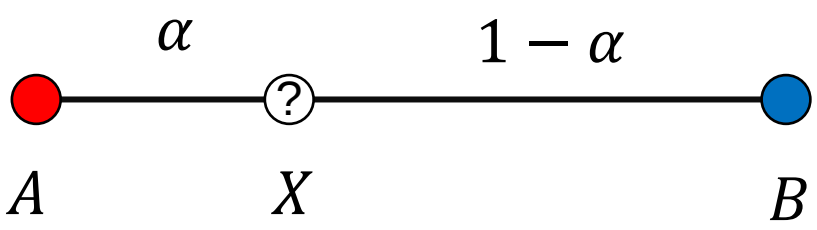
u_i=?



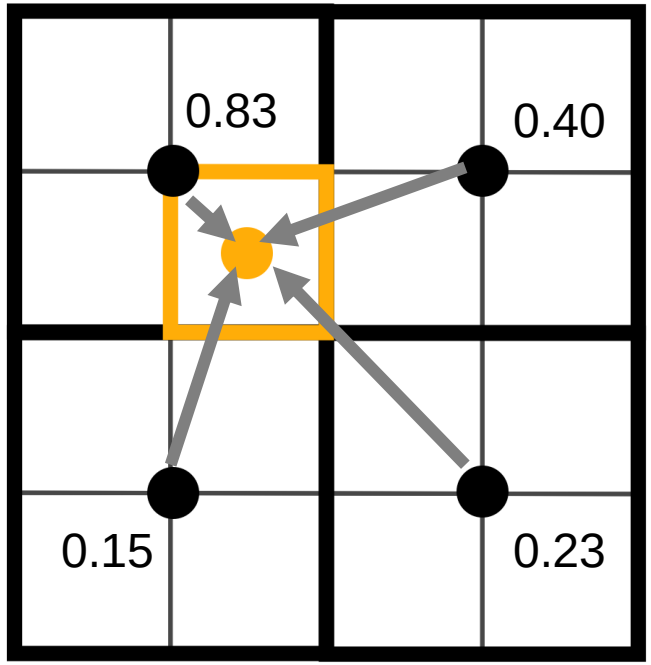
(Bi)linear interpolation

Linear interpolation (1D)

$$f(X) = ?$$



Bi-linear interpolation (2D)



(Bi)linear interpolation

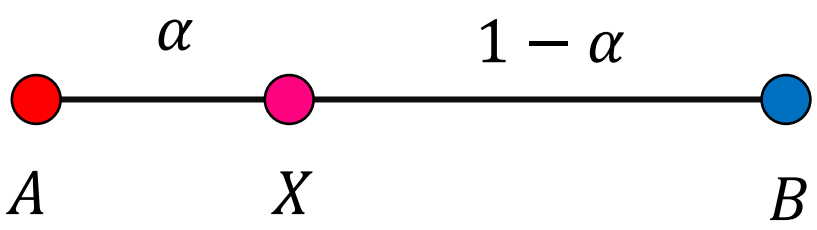
$$\frac{f(X) - f(A)}{\alpha} = \frac{f(B) - f(A)}{\alpha + (1 - \alpha)}$$

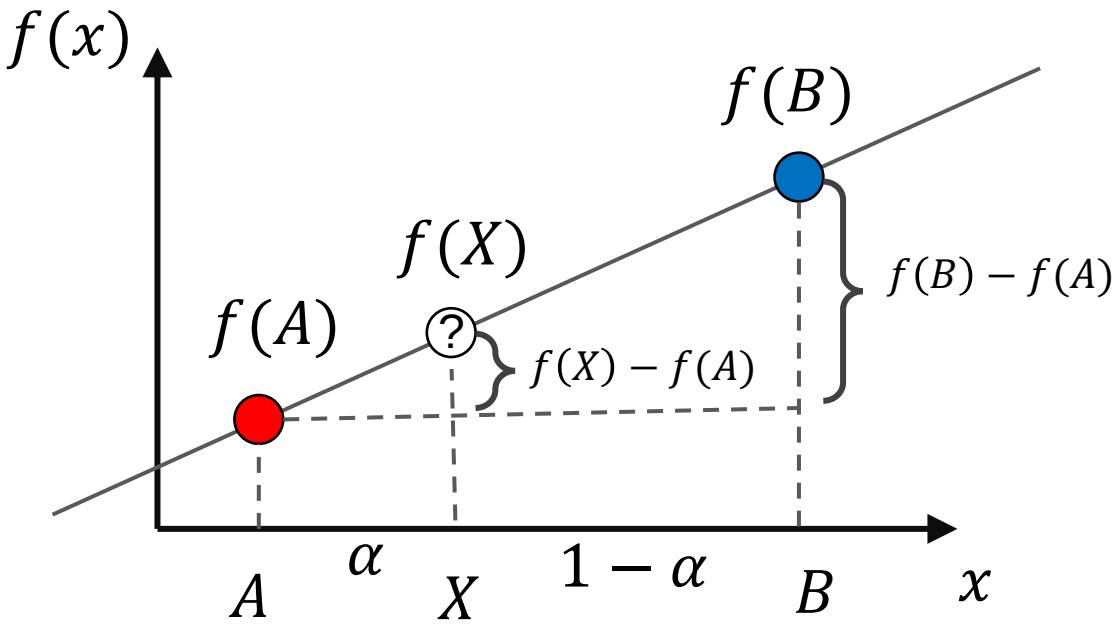
$$f(X) - f(A) = \big(f(B) - f(A)\big)\alpha$$

$$f(X) = f(A) \cdot (1 - \alpha) + f(B)\alpha$$

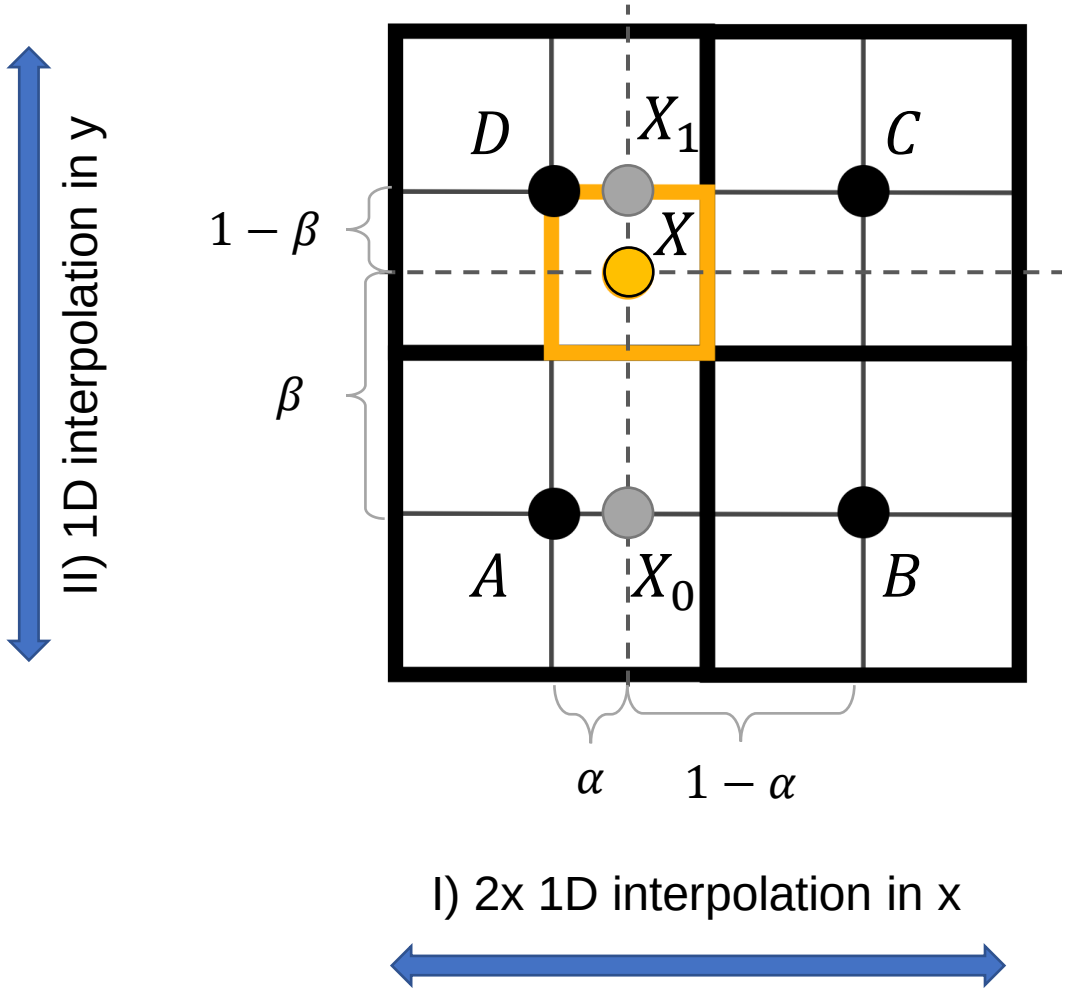
Linear interpolation (1D)

$$f(X) = f(A) \cdot (1 - \alpha) + f(B) \cdot \alpha$$





(Bi)linear interpolation $f(X) = f(A) \cdot (1-\gamma) + f(B) \cdot \gamma$



I) $f(X_0) = f(A) \cdot (1-\alpha) + f(B) \cdot \alpha$

$f(X_1) = f(D) \cdot (1-\alpha) + f(C) \cdot \alpha$

II) $f(X) = f(X_0) \cdot (1-\beta) + f(X_1) \cdot \beta$

For $\alpha = \frac{1}{4}$ and $\beta = \frac{3}{4}$

$f(A)$	$(1-\alpha)(1-\beta)$	$3/16$
$f(B)$	$(\alpha)(1-\beta)$	$1/16$
$f(C)$	$(\alpha)(\beta)$	$3/16$
$f(D)$	$(1-\alpha)(\beta)$	$9/16$

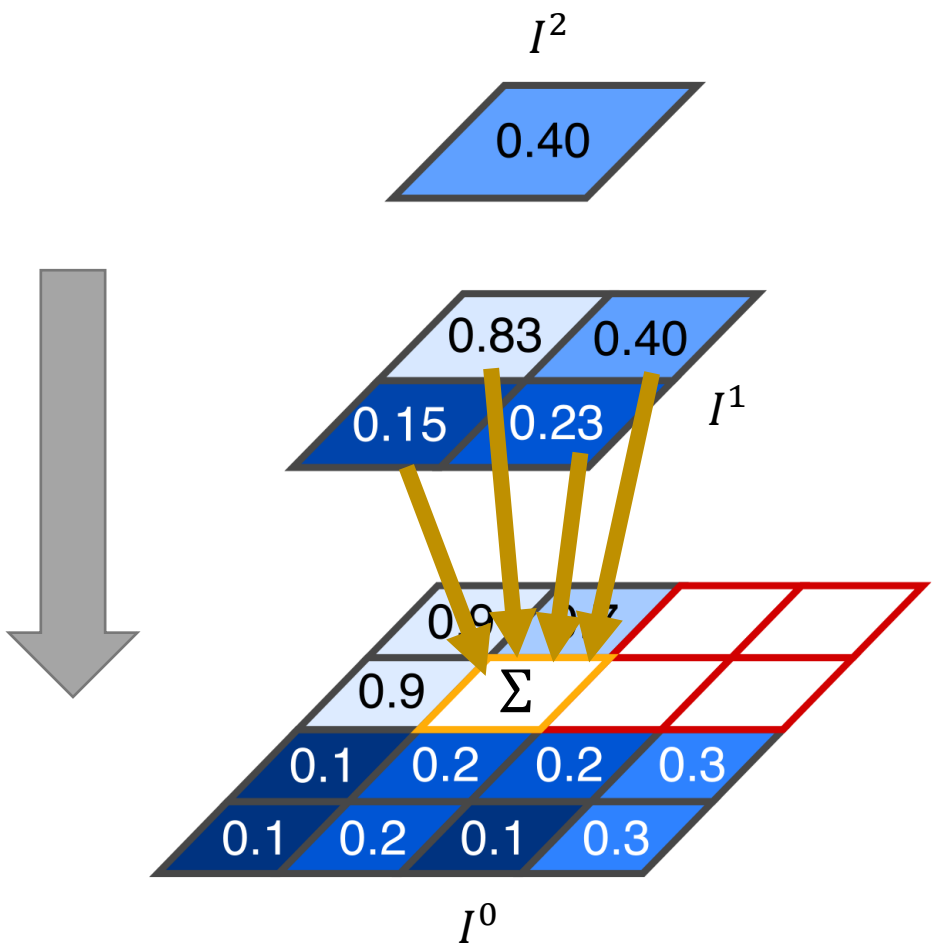
$$u_i \in \left\{\frac{1}{16}, \frac{3}{16}, \frac{3}{16}, \frac{9}{16}\right\}$$

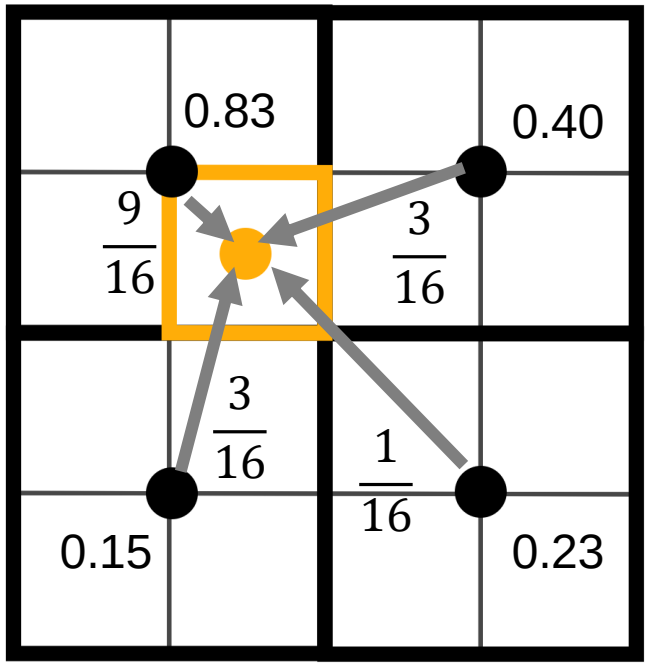
2) Push

Interpolate 4 samples from level $k+1$.

$$I^k = \sum_{i=0}^{4} I_i^{k+1} u_i \qquad \mathrm{u_i} \in \left\{\frac{1}{16}, \frac{3}{16}, \frac{3}{16}, \frac{9}{16}\right\}$$

$$I^k = 0.23 \cdot \frac{1}{16} + 0.15 \cdot \frac{3}{16} + 0.40 \cdot \frac{3}{16} + 0.83 \cdot \frac{9}{16} \approx 0.58$$



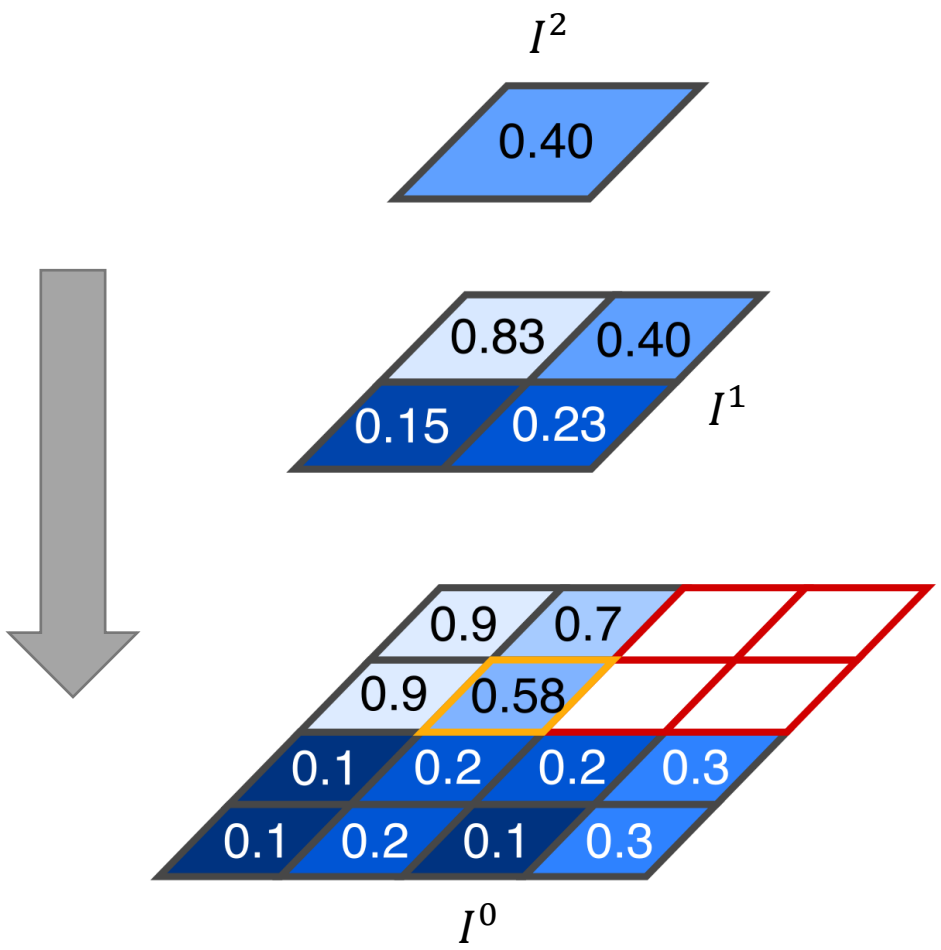


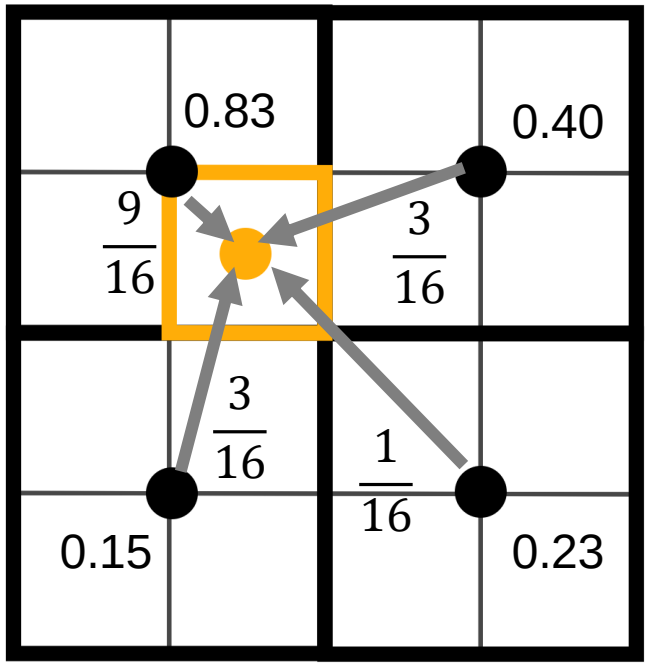
2) Push

Interpolate 4 samples from level $k+1$.

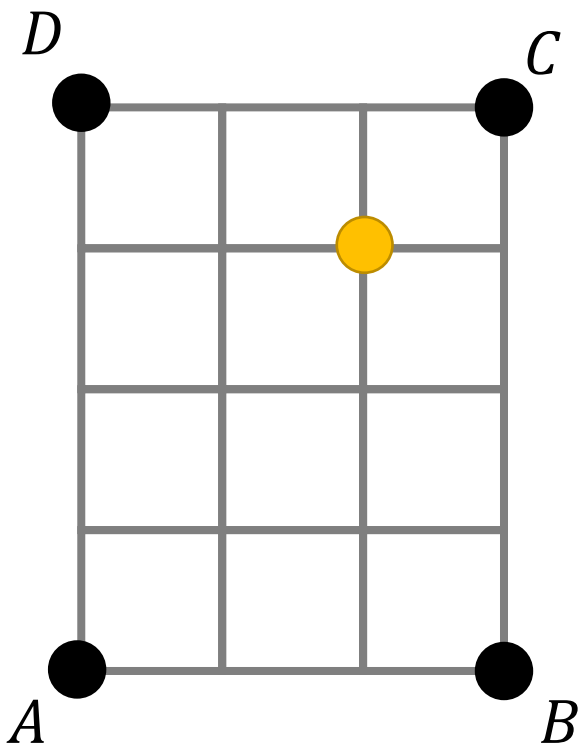
$$I^k = \sum_{i=0}^{4} I_i^{k+1} u_i \qquad \mathrm{u_i} \in \left\{\frac{1}{16}, \frac{3}{16}, \frac{3}{16}, \frac{9}{16}\right\}$$

$$I^k = 0.23 \cdot \frac{1}{16} + 0.15 \cdot \frac{3}{16} + 0.40 \cdot \frac{3}{16} + 0.83 \cdot \frac{9}{16} \approx 0.58$$





What are the correct bilinear weights?



A

$$u_A = \frac{1}{16}$$

$$u_C = \frac{9}{16}$$

B

$$u_A = \frac{6}{12}$$

$$u_C = \frac{1}{12}$$

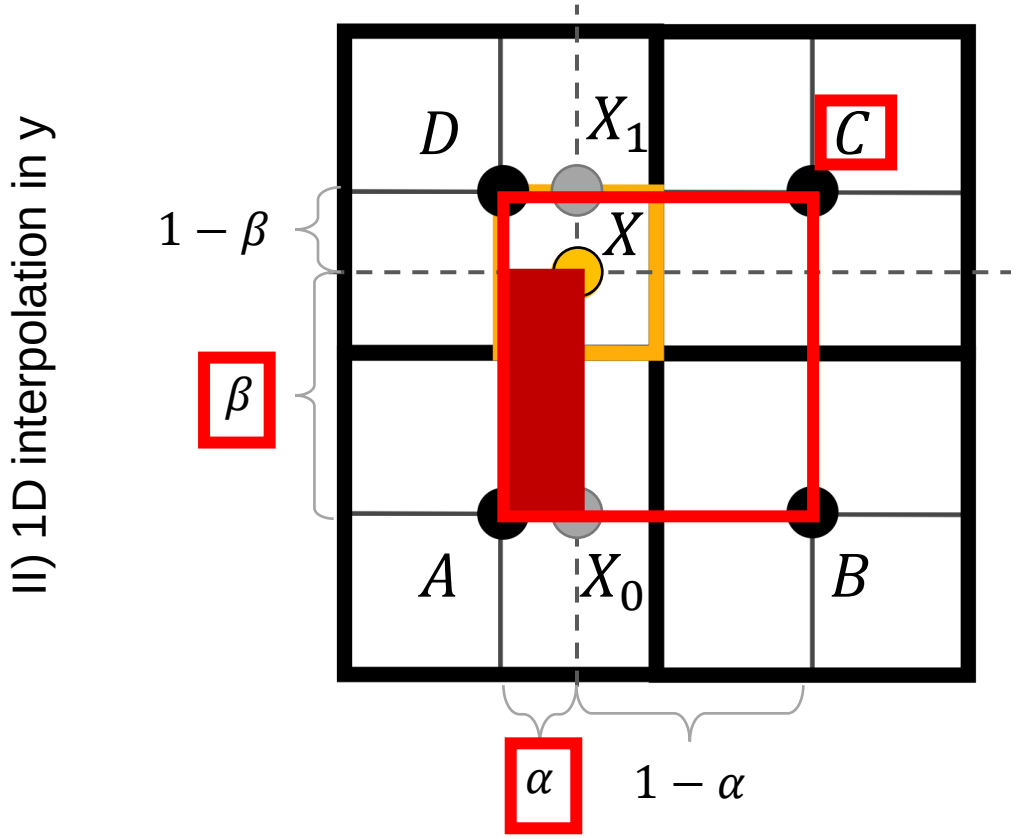
C

$$u_A = \frac{1}{12}$$

$$u_C = \frac{6}{12}$$

Do not compute, estimate!

(Bi)linear interpolation



I) 2x 1D interpolation in x

I) $f(X_0) = f(A)\cdot(1-\alpha) + f(B)\cdot\alpha$

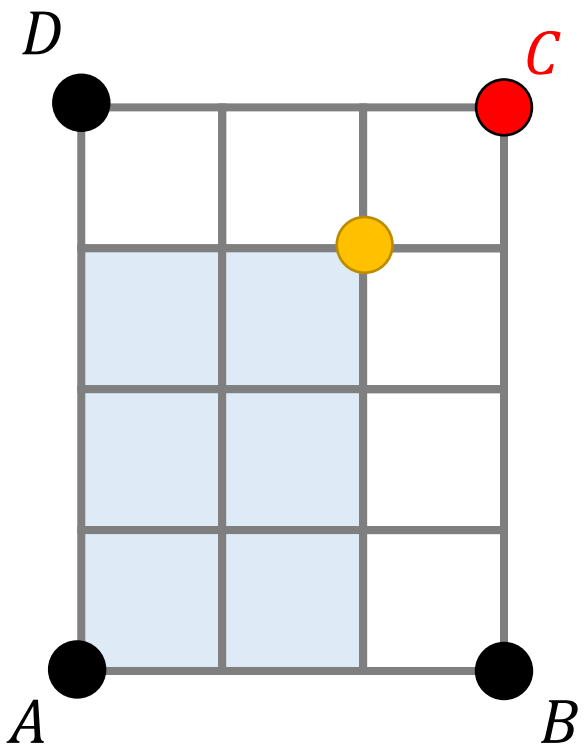
$f(X_1) = f(D)\cdot(1-\alpha) + f(C)\cdot\alpha$

II) $f(X) = f(X_0)\cdot(1-\beta) + f(X_1)\cdot\beta$

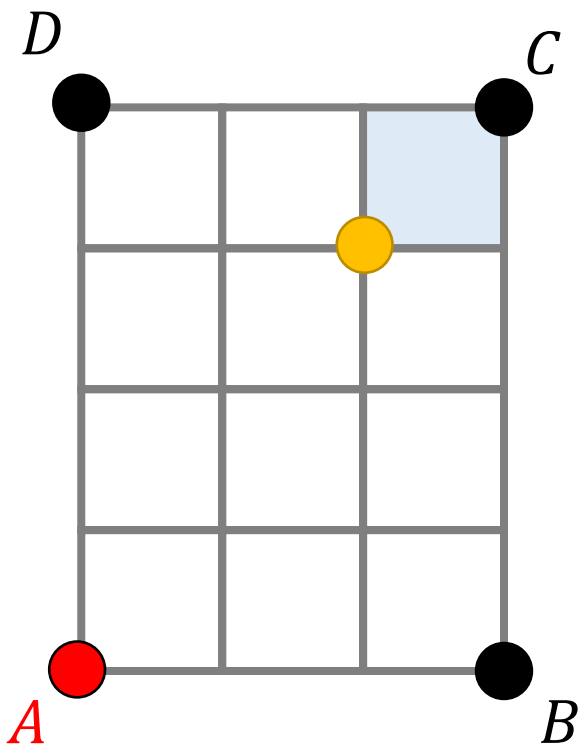
For $\alpha = \frac{1}{4}$ and $\beta = \frac{3}{4}$

$f(A)$	$(1-\alpha)(1-\beta)$	$3/16$
$f(B)$	$(\alpha)(1-\beta)$	$1/16$
$f(C)$	$(\alpha)(\beta)$	$3/16$
$f(D)$	$(1-\alpha)(\beta)$	$9/16$

$$u_i \in \left\{\frac{1}{16}, \frac{3}{16}, \frac{3}{16}, \frac{9}{16}\right\}$$

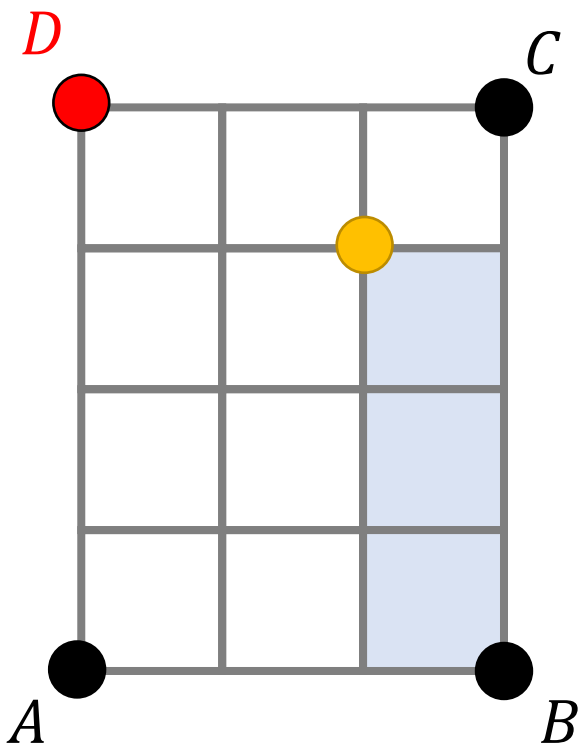


$$u_C = \frac{6}{12}$$



$$u_A = \frac{1}{12}$$

$$u_C = \frac{6}{12}$$



$$u_A = \frac{1}{12}$$

$$u_B = \frac{2}{12}$$

$$u_C = \frac{6}{12}$$

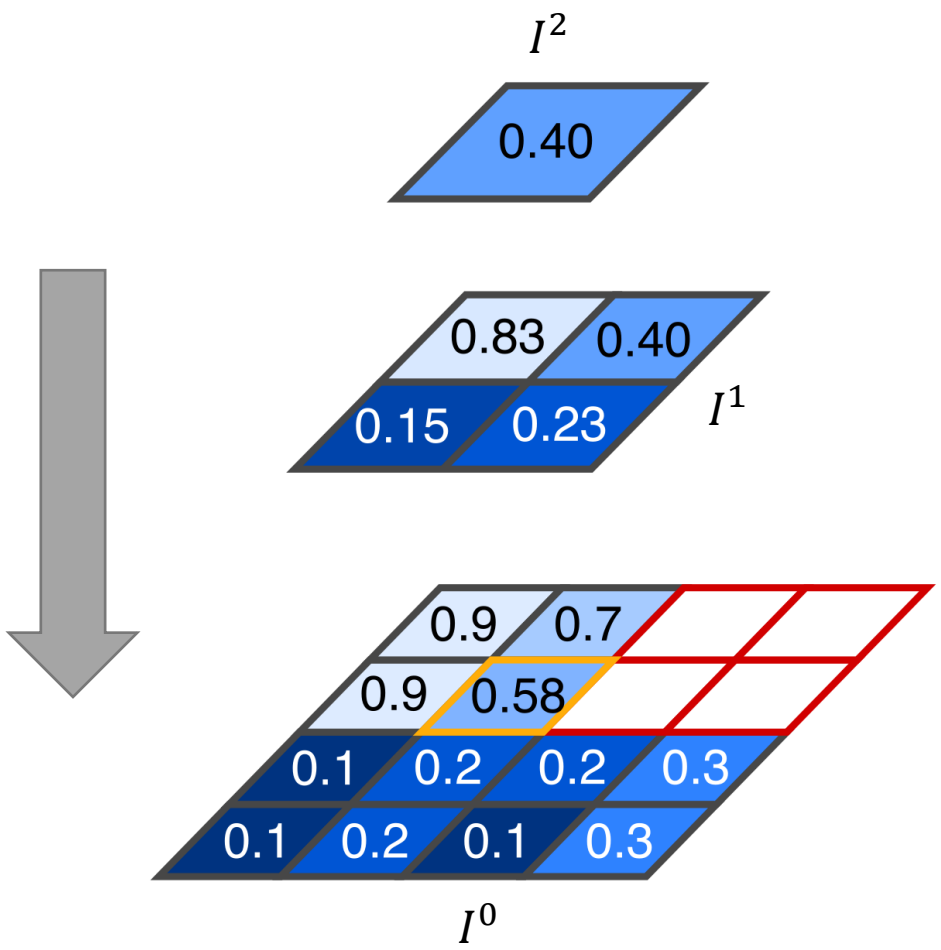
$$u_D = \frac{3}{12}$$

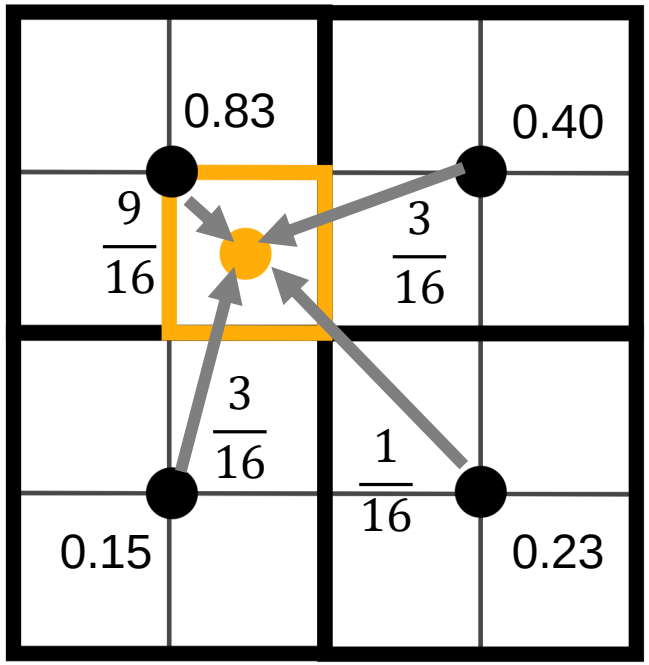
2) Push

Interpolate 4 samples from level $k+1$.

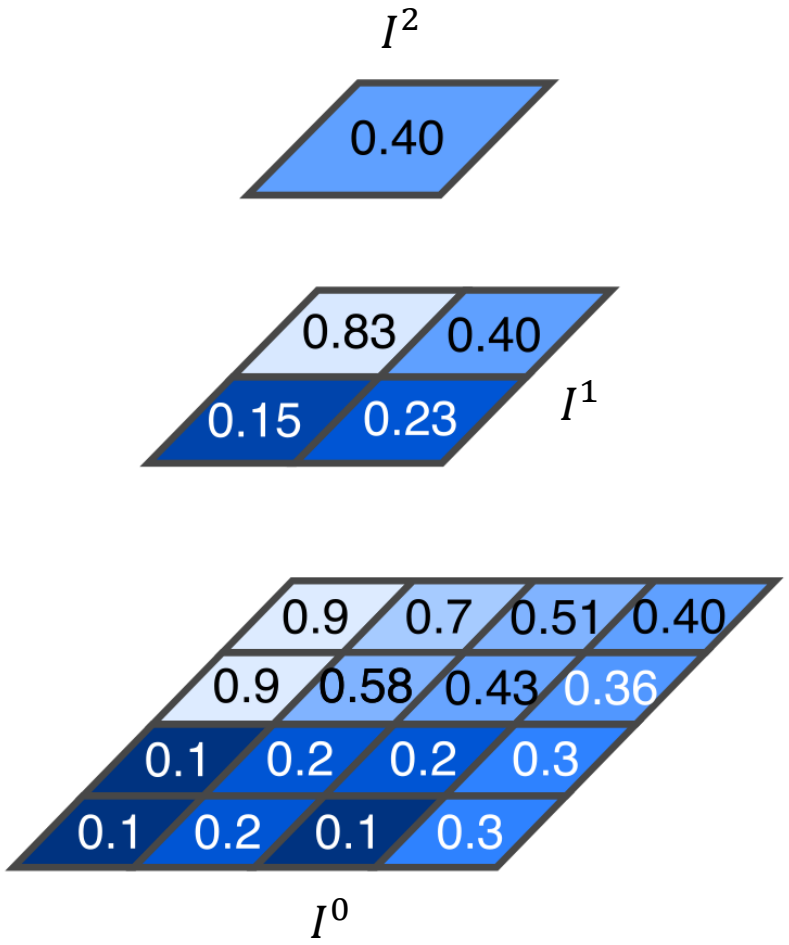
$$I^k = \sum_{i=0}^{4} I_i^{k+1} u_i \qquad \mathrm{u_i} \in \left\{\frac{1}{16}, \frac{3}{16}, \frac{3}{16}, \frac{9}{16}\right\}$$

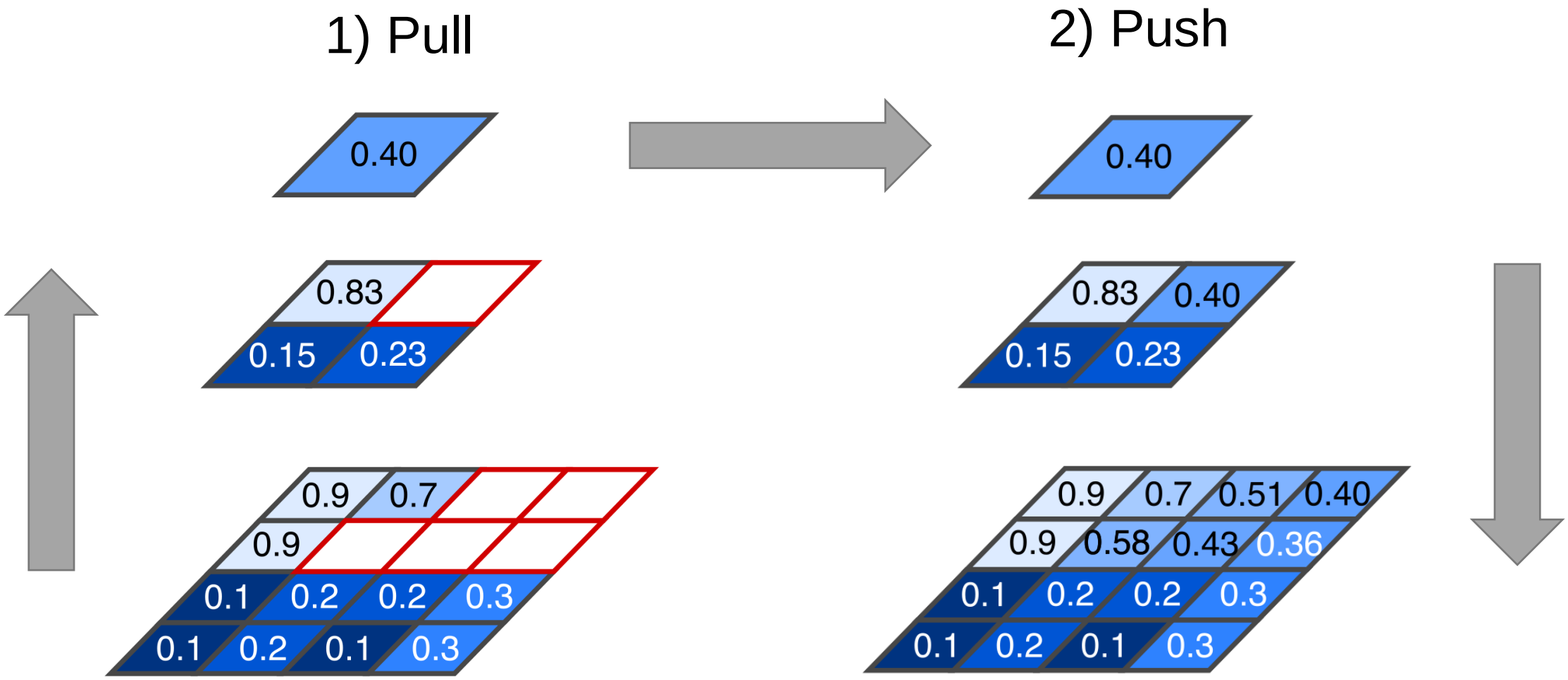
$$I^k = 0.23 \cdot \frac{1}{16} + 0.15 \cdot \frac{3}{16} + 0.40 \cdot \frac{3}{16} + 0.83 \cdot \frac{9}{16} \approx 0.58$$





2) Push


I^2
0.40
0.83
0.40
0.15
0.23
I^1
0.9
0.7
0.51
0.40
0.9
0.58
0.43
0.36
0.1
0.2
0.2
0.3
0.1
0.2
0.1
0.3
I^0



Aggregate information.

Distribute information.

Input image

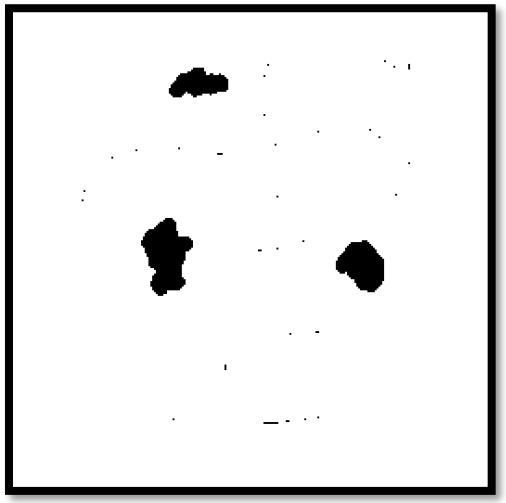
Image credit: jtrend @ flickr.com

Weights

1) **Pull** (step 1)

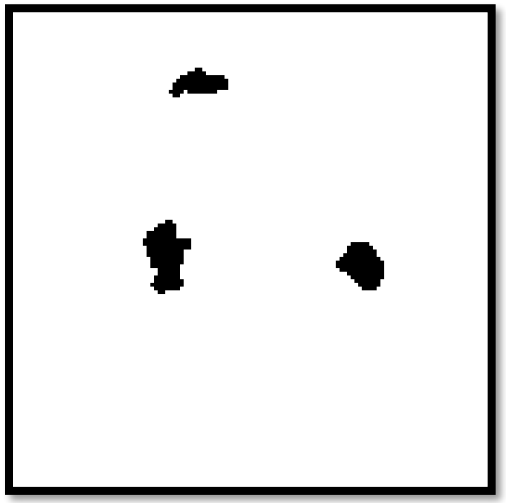
Image credit: jtrend @ flickr.com

Weights



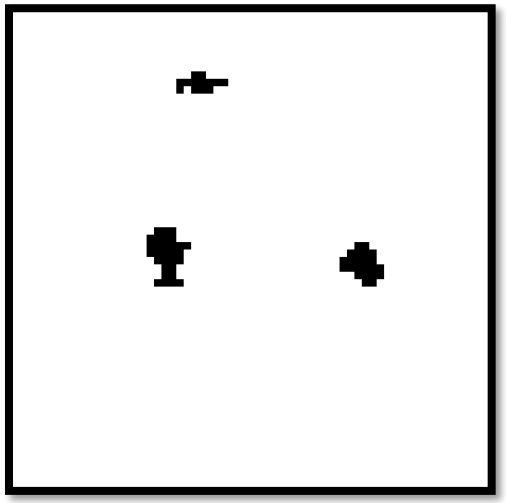
1) **Pull** (step 2)

Weights

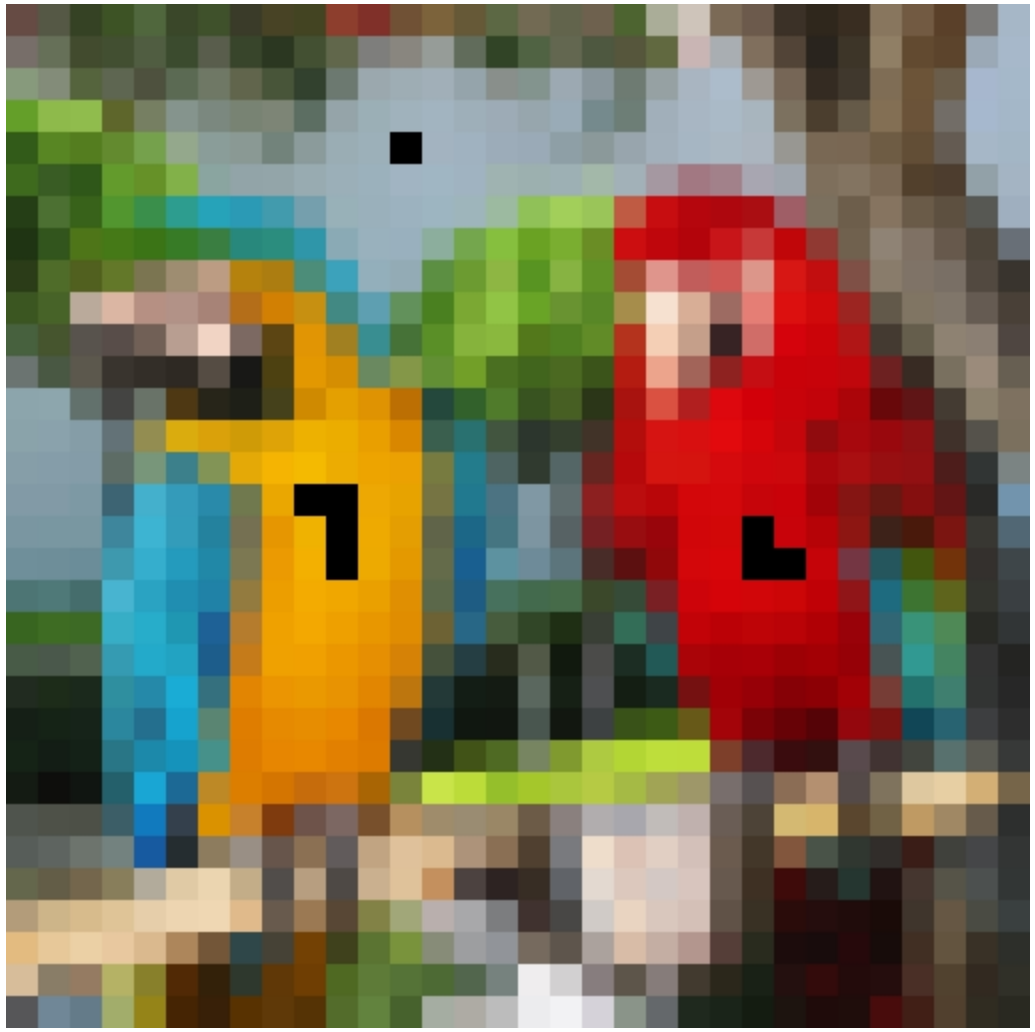


1) **Pull** (step 3)

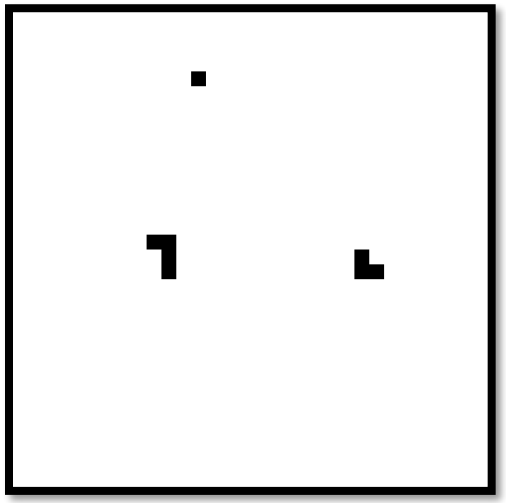
Weights



1) **Pull** (step 4)

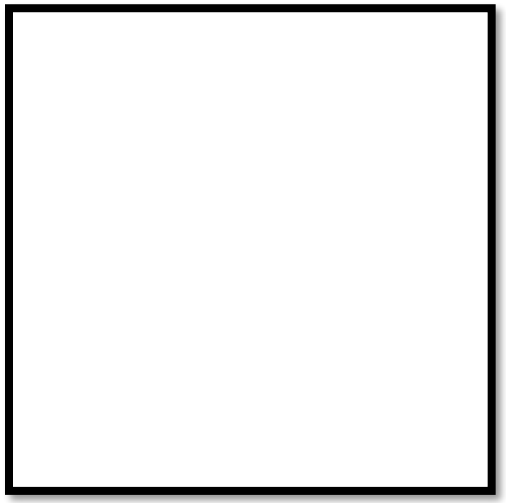


Weights



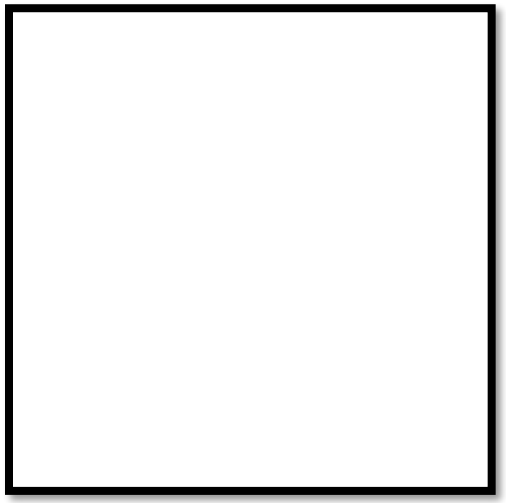
1) **Pull** (step 5)

Weights



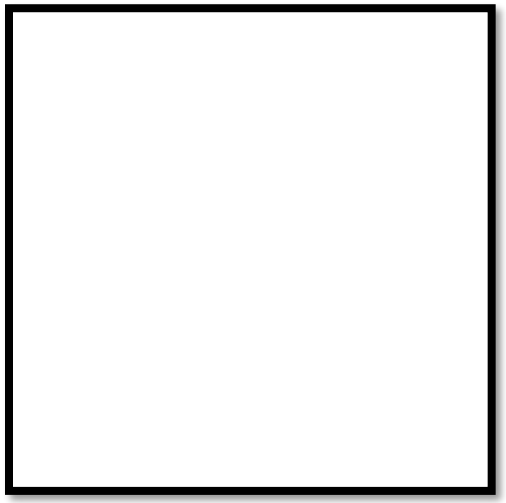
1) **Pull** (step 6)

Weights

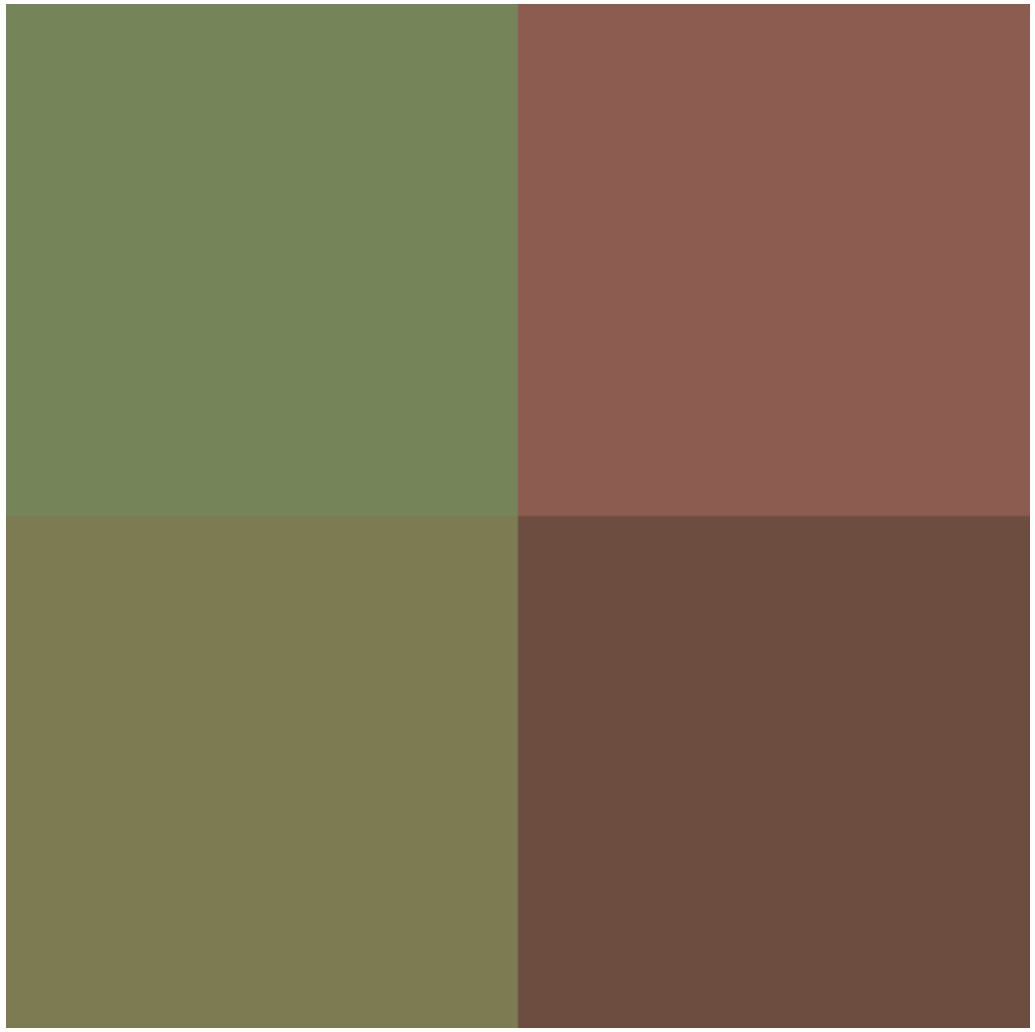


1) **Pull** (step 7)

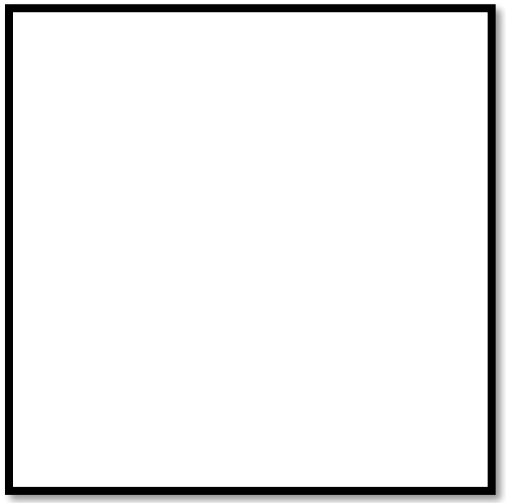
Weights



1) **Pull** (step 8)

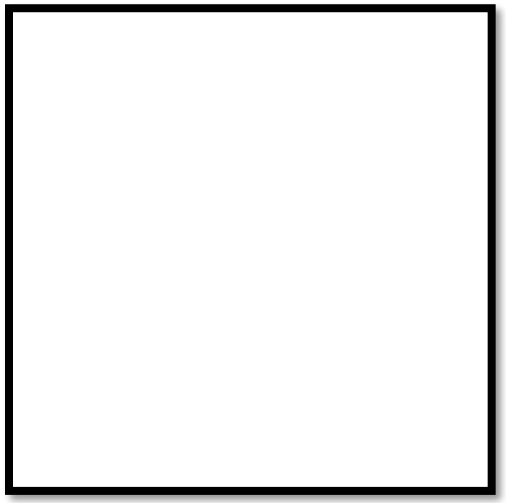


Weights

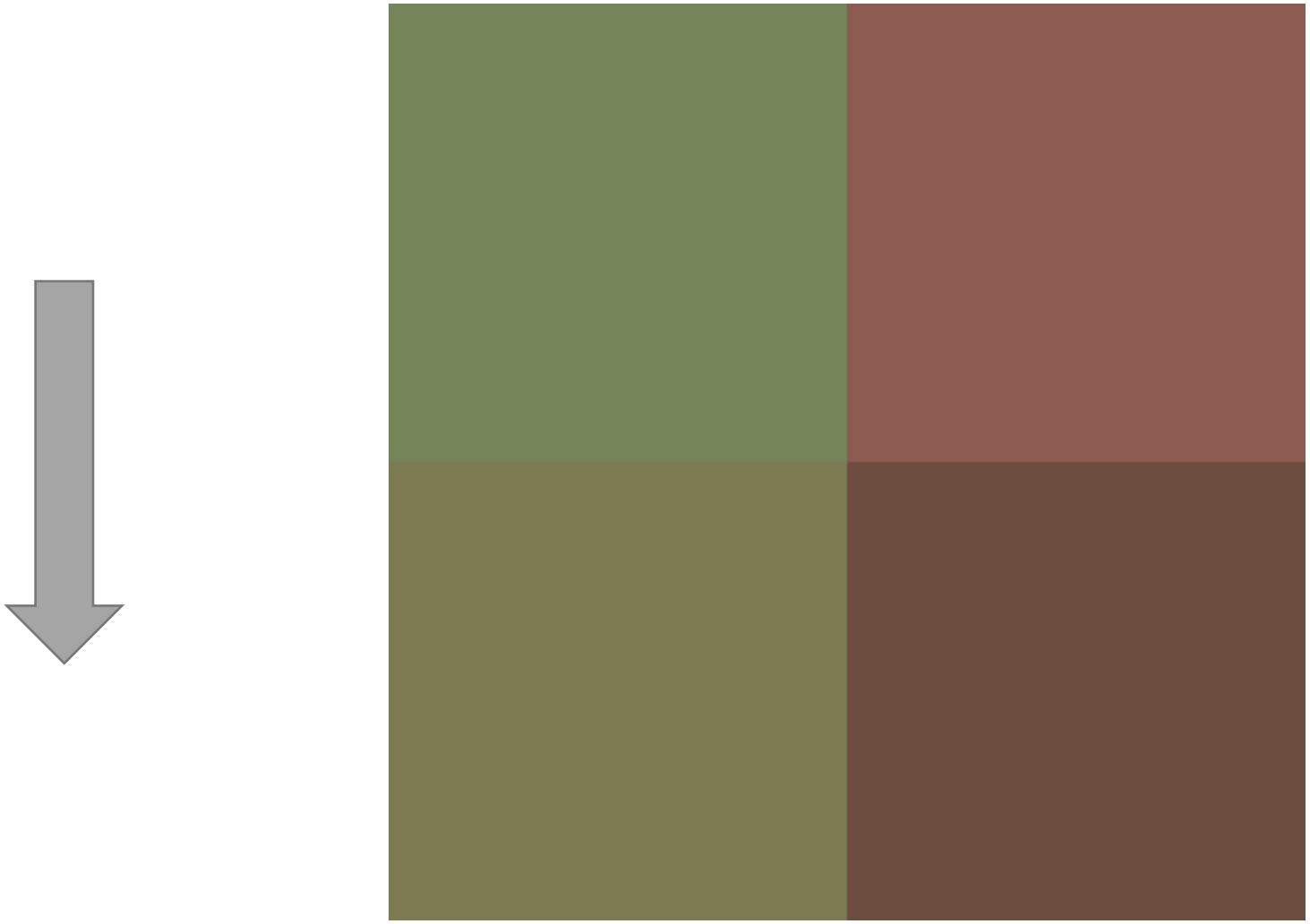


1) **Pull** (step 9)

Weights



2) **Push** (step 8)



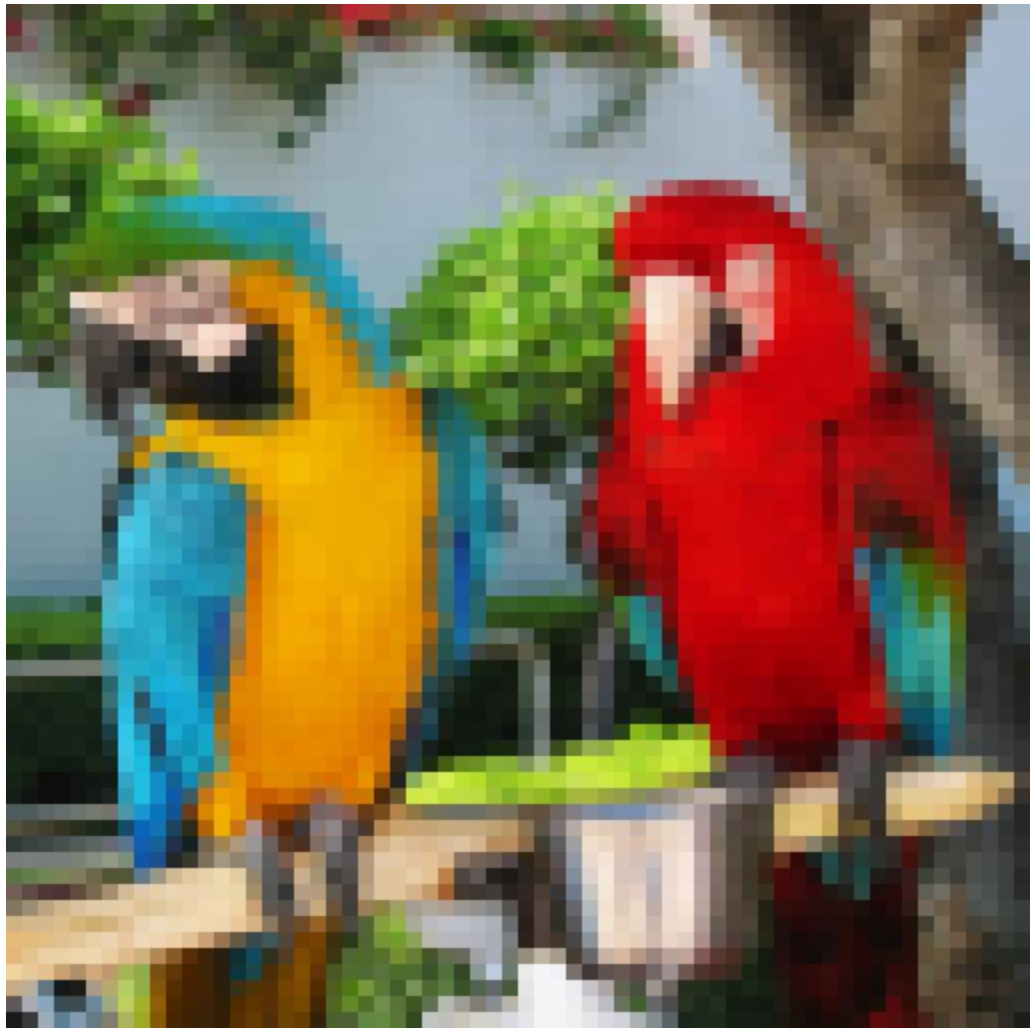
2) **Push** (step 7)

2) **Push** (step 6)

2) **Push** (step 5)

2) **Push** (step 4)

2) **Push** (step 3)



2) **Push** (step 2)

2) **Push** (step 1)

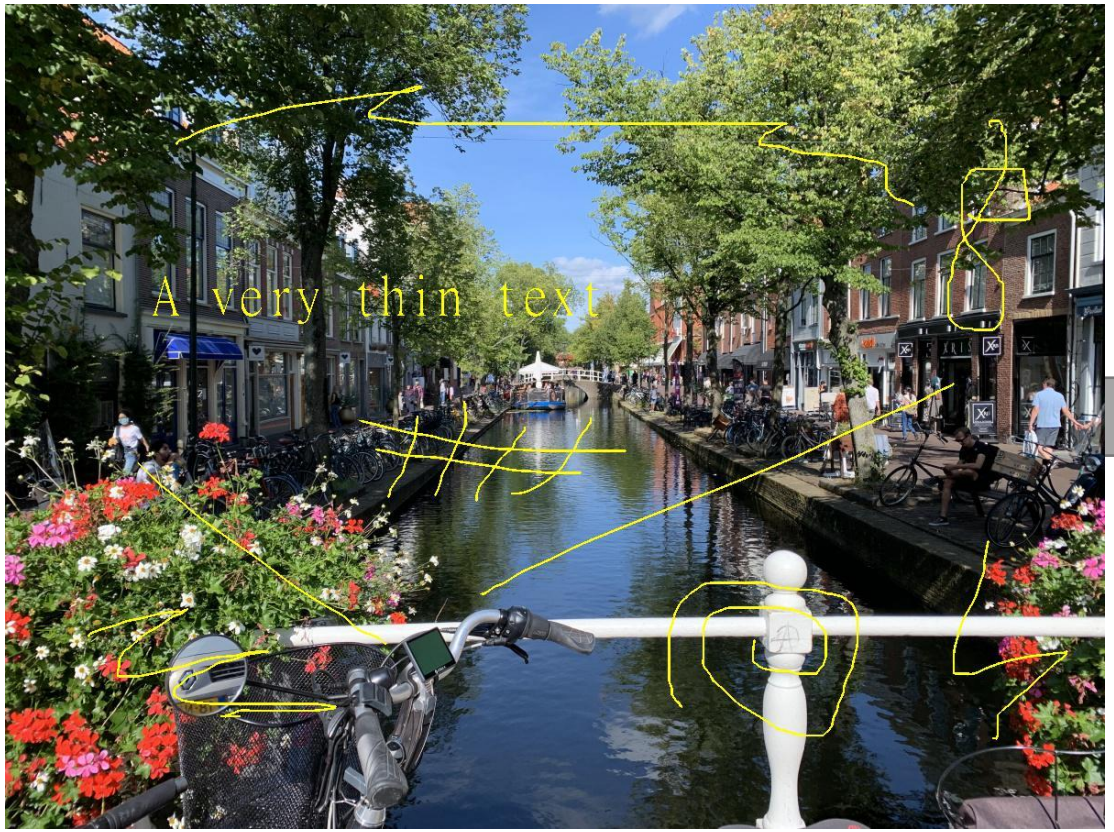
2) **Push** (step 0)

Input

Result

Examples

Good results for small/thin missing regions.

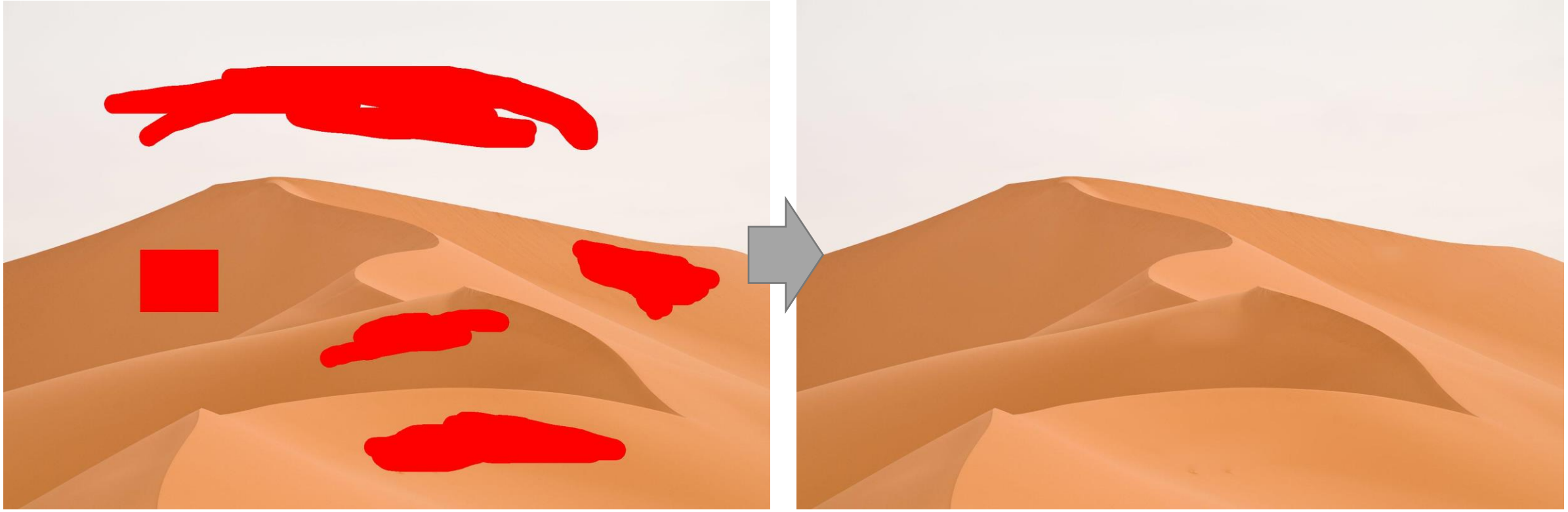


Input image

Inpainted image

Examples

Works well for even large uniform regions.



Hembo Pagi @ flickr.com

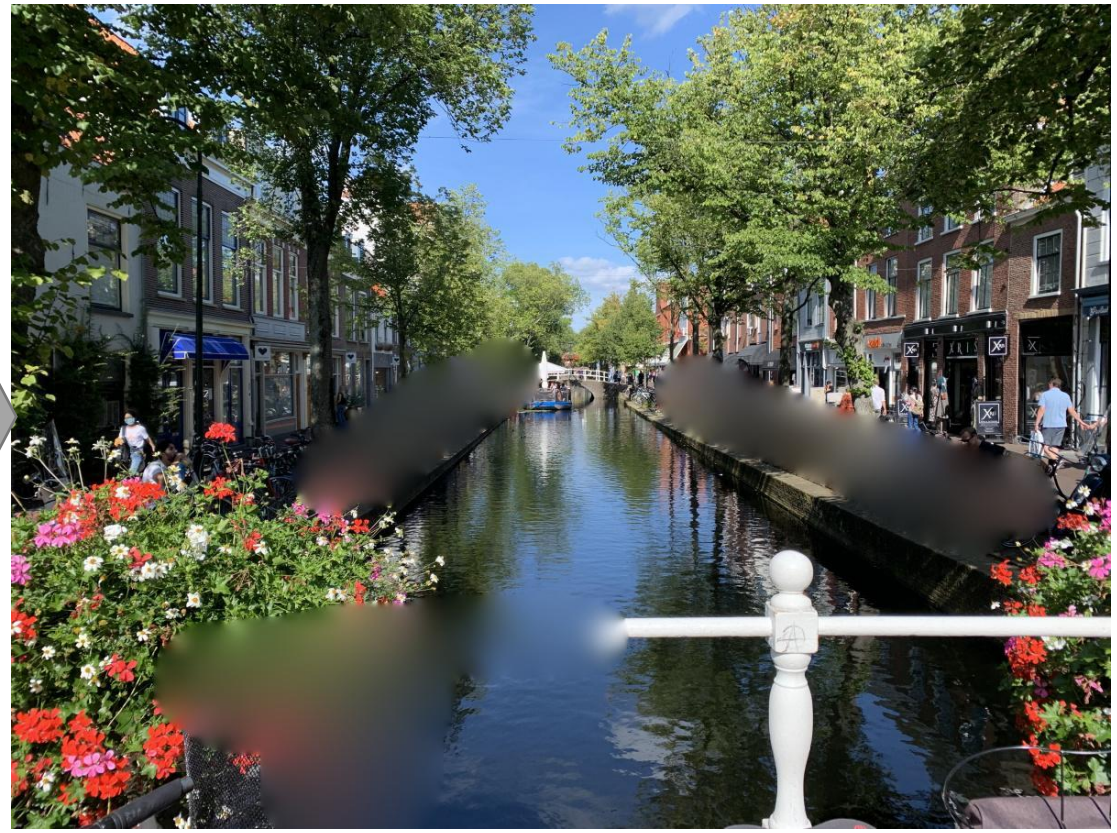
Input image

Inpainted image

Examples

Cannot handle large textured regions.

Input image



Inpainted image

Pull-Push

Good

- Easy to implement.
- Fast to run.

Bad

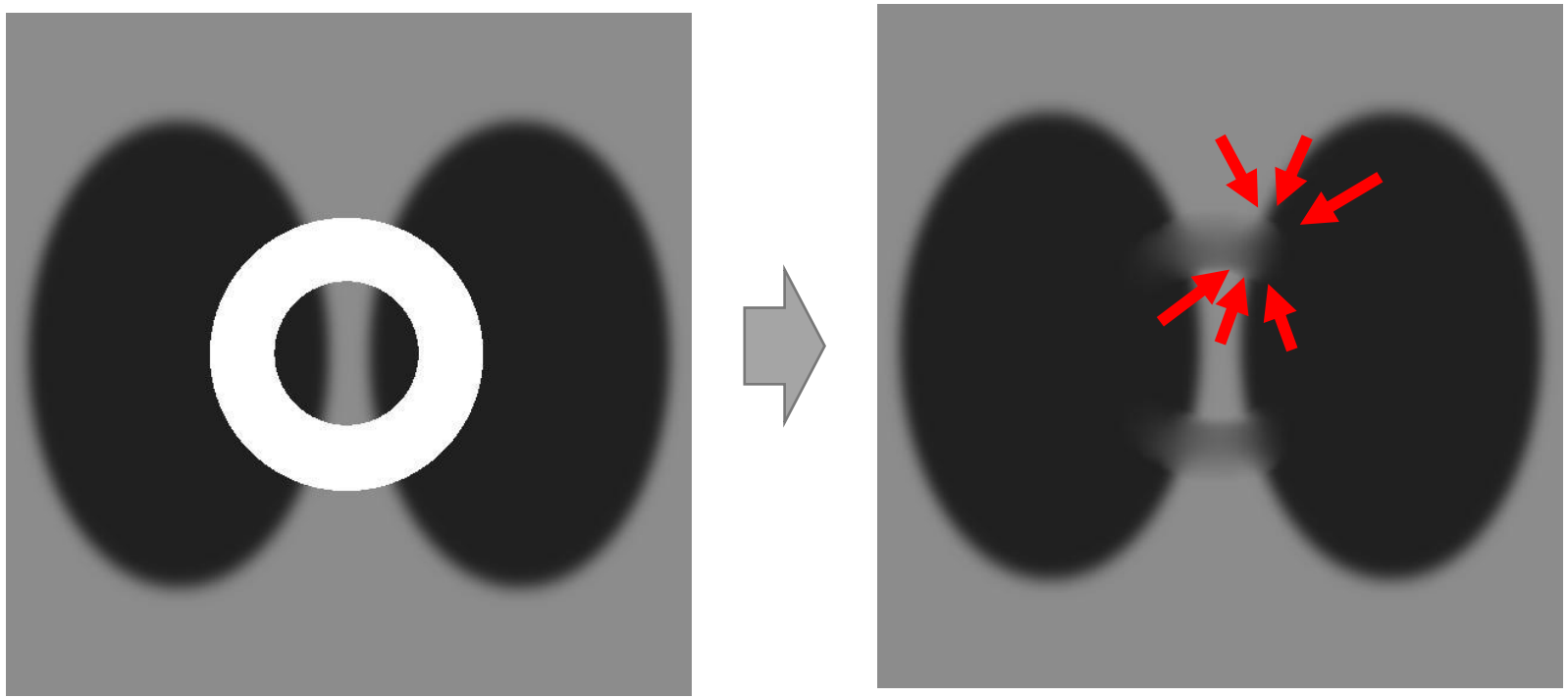
- Cannot produce any texture.
- Usually, suitable only for small regions.

Algorithms

	Statistic		Prior
1. Pull-Push	Simple: Mean color		Single image
2. PDE inpainting	Simple: Gradients		Single image
3. Exemplar inpainting	Medium: Patterns		Single image
4. Collection sampling	Medium: Patterns		Collection
5. Neural inpainting	Complex: Multi-scale		Collection

Motivation

Pull push does not respect direction of edges in the image.

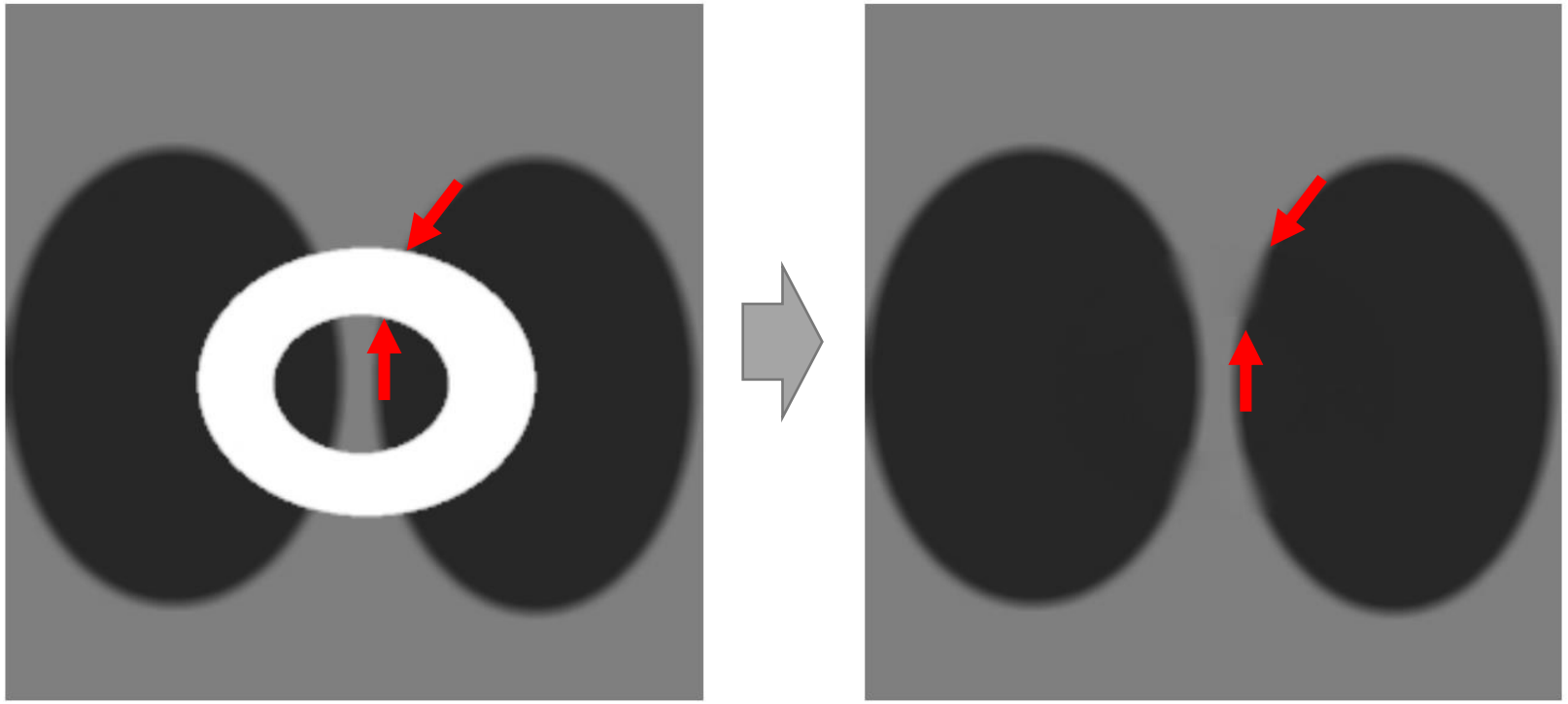


Input image

Inpainted image

Inpainting using Partial Differential Equations (PDE)

- **Idea**: Follow image **isophotes**.



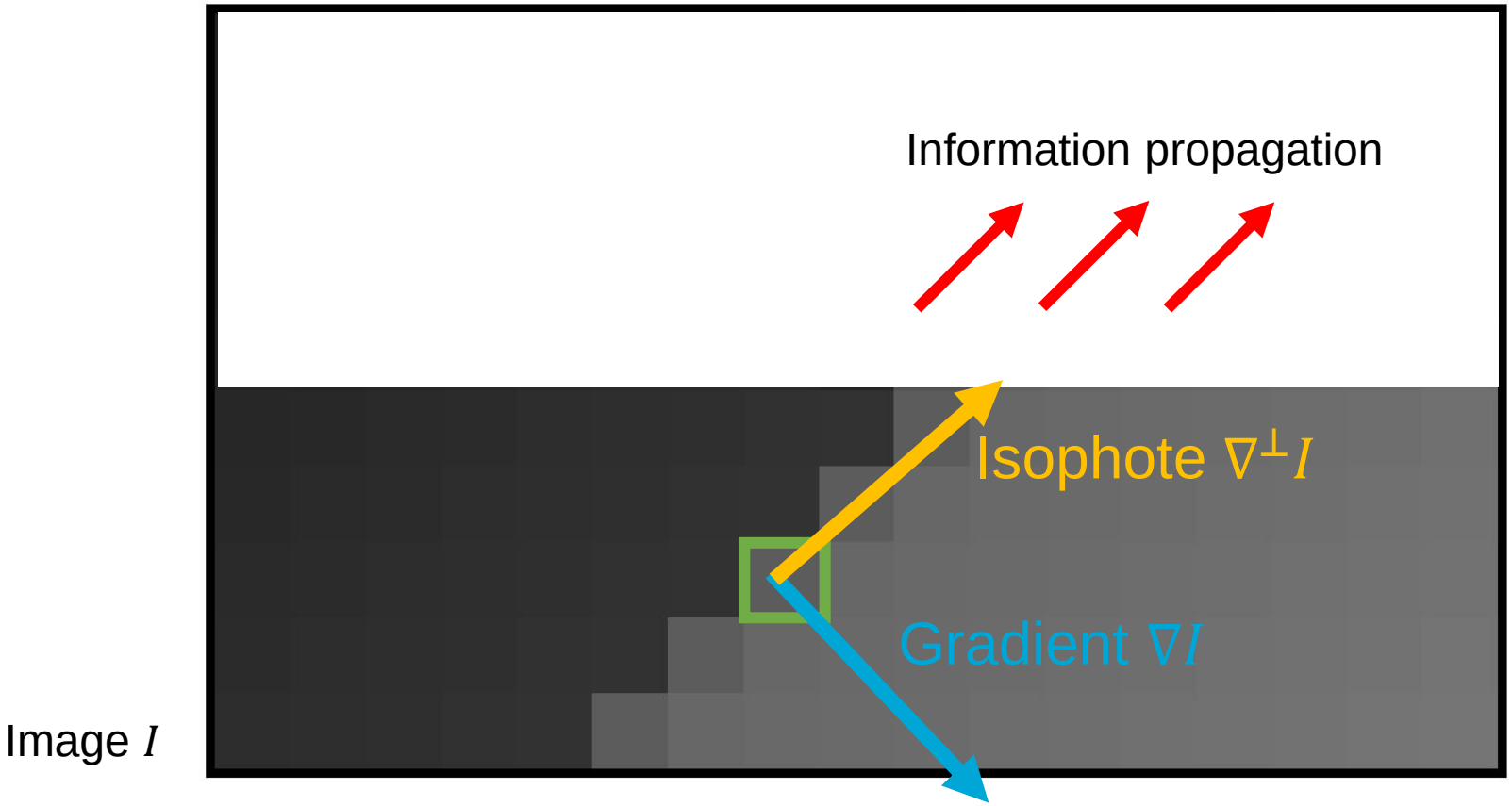
Input image

Inpainted image

Bertalmio, Marcelo, et al. "Image inpainting." Proceedings of the 27th annual conference on Computer graphics and interactive techniques. 2000.

Isophote

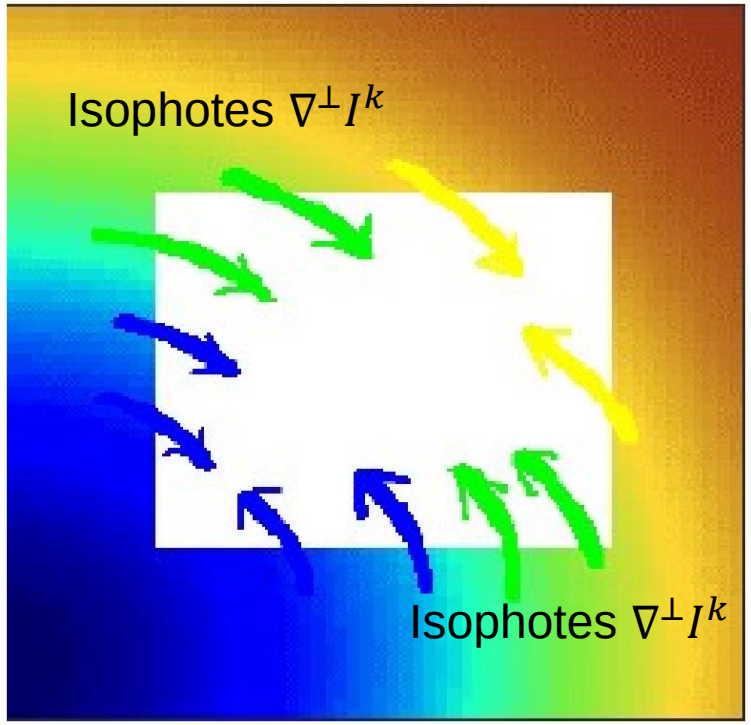
= Curve/path of pixels with equal brightness/color.



Bertalmio, Marcelo, et al. "Image inpainting." Proceedings of the 27th annual conference on Computer graphics and interactive techniques. 2000.

Iterative inpainting

Iteratively apply the update rule until convergence.



Information propagation

Update rule:

$$\underbrace{I^{k+1}}_{\text{Next image}} = \underbrace{I^k}_{\text{Current image}} + \alpha \cdot \underbrace{\left(\nabla L^k \cdot \nabla^{\perp} I^k\right)}_{\text{Flow of information along the isophote}}$$

Very similar to Gradient based methods from Lecture 3.

Bertalmio, Marcelo, et al. "Image inpainting." Proceedings of the 27th annual conference on Computer graphics and interactive techniques. 2000.

Iterative inpainting

Step #0

Time →

Step #N

Bertalmio, Marcelo, et al. "Image inpainting." Proceedings of the 27th annual conference on Computer graphics and interactive techniques. 2000.

Examples

Input image

Inpainted image

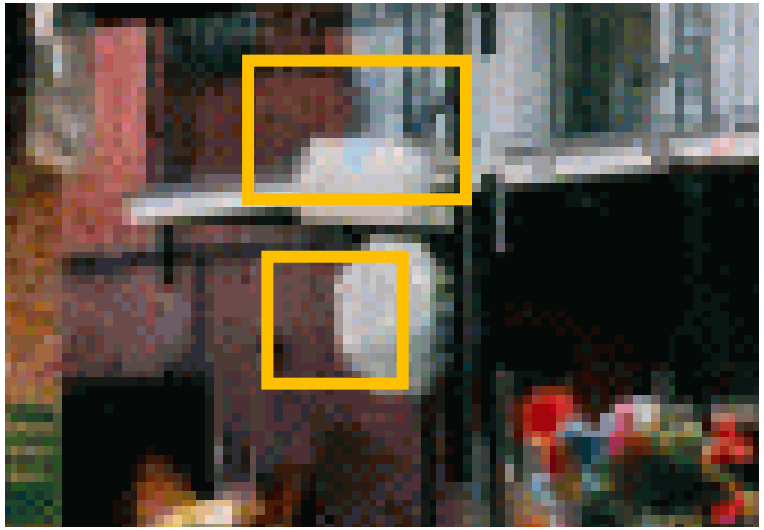
Bertalmio, Marcelo, et al. "Image inpainting." Proceedings of the 27th annual conference on Computer graphics and interactive techniques. 2000.

Comparison to Pull-Push

Reduces blurring of the image in presence of strong edges.

Input image

Pull-push

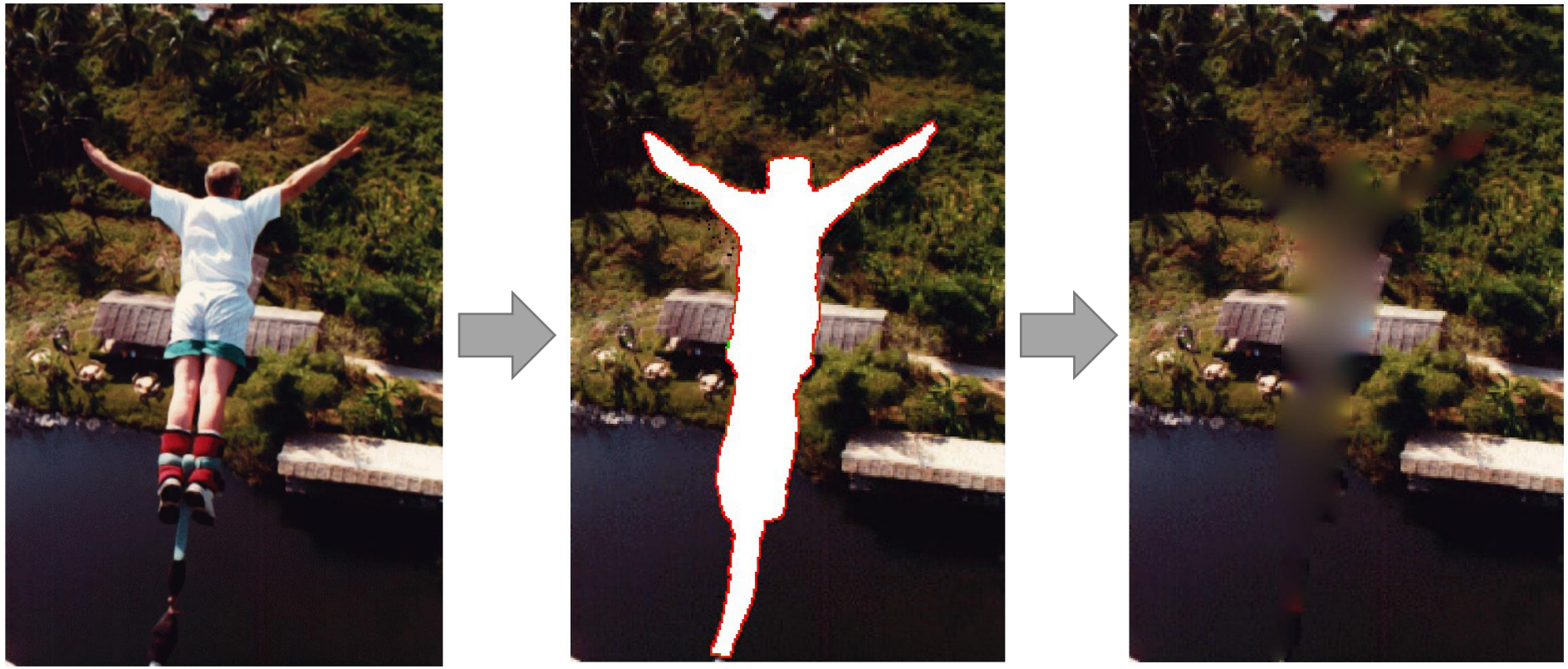


PDE

Bertalmio, Marcelo, et al. "Image inpainting." Proceedings of the 27th annual conference on Computer graphics and interactive techniques. 2000.

Limitations

Still cannot fill large areas with complex structures.



Input image | Removed pixels | Result

Criminisi, Antonio, Patrick Pérez, and Kentaro Toyama. "Region filling and object removal by exemplar-based image inpainting." IEEE TIP (2004).

PDE Inpainting

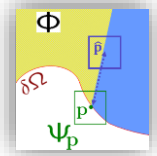
Good

- Preserves edges.
- Still relatively simple.

Bad

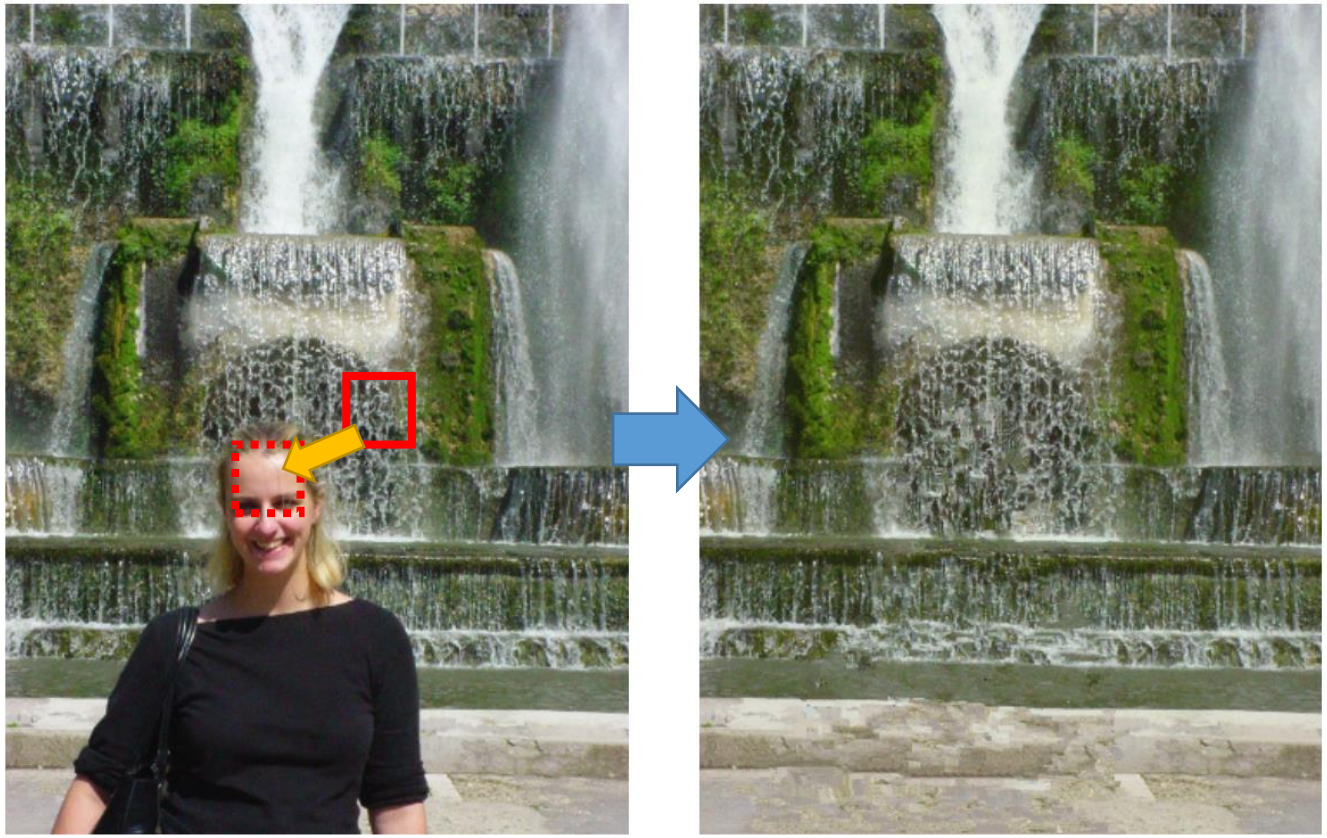
- Still no textures which limits application to only small regions.

Algorithms

	Statistic		Prior
1. Pull-Push	Simple: Mean color		Single image
2. PDE inpainting	Simple: Gradients		Single image
3. Exemplar inpainting	Medium: Patterns		Single image
4. Collection sampling	Medium: Patterns		Collection
5. Neural inpainting	Complex: Multi-scale		Collection

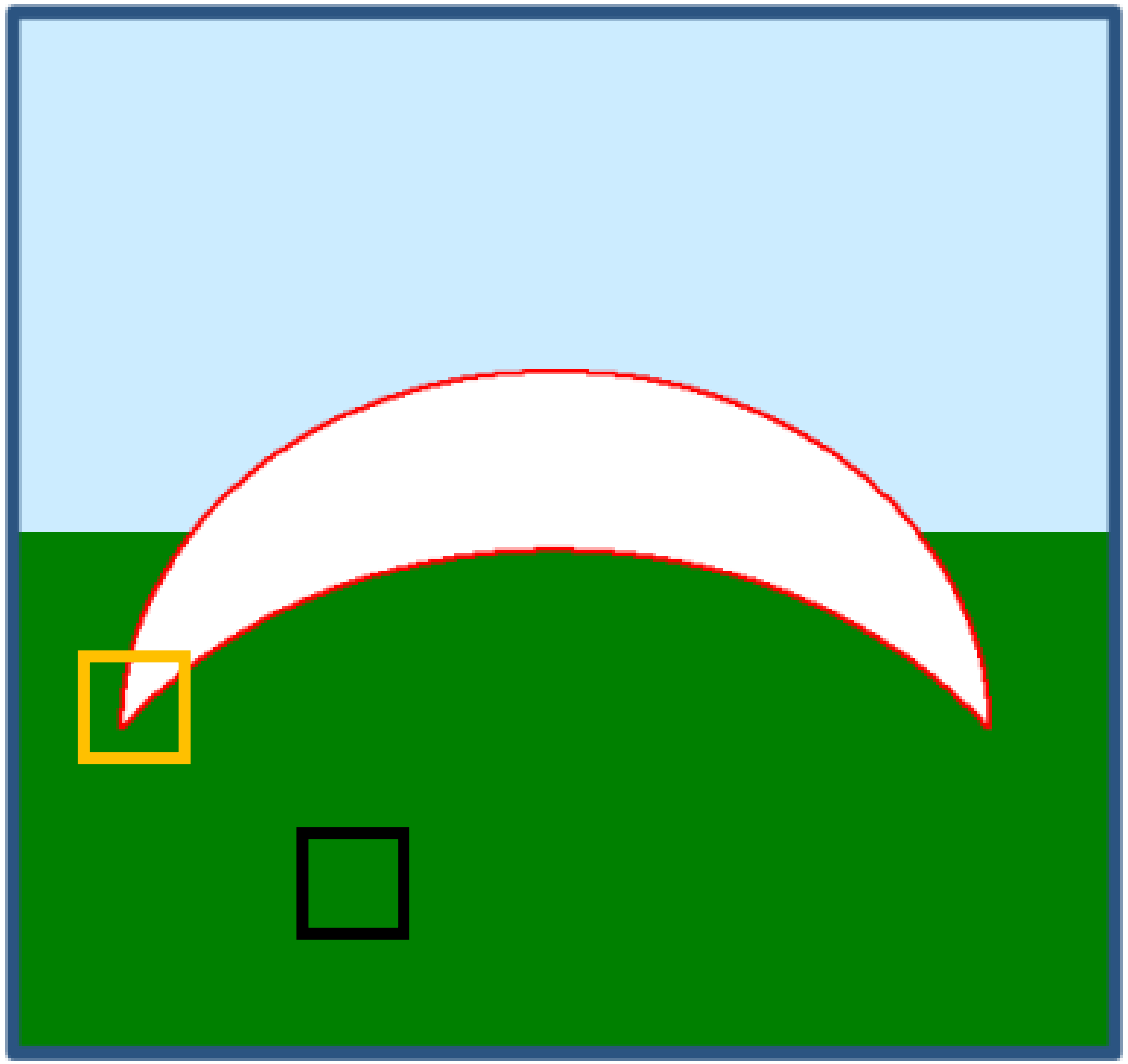
Exemplar-based image inpainting

Idea: Clone content from another part of the image.

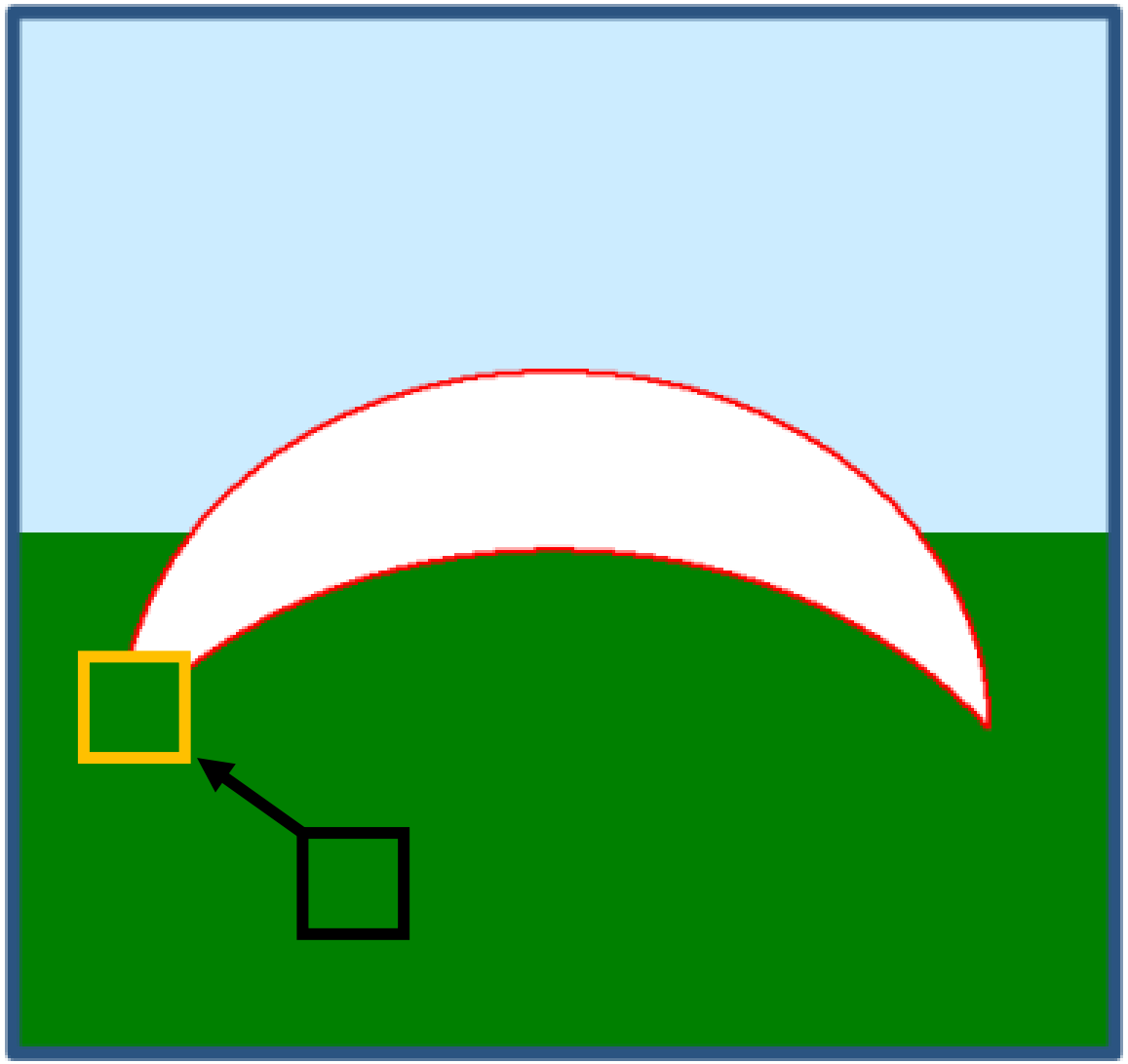


Criminisi, Antonio, Patrick Pérez, and Kentaro Toyama. "Region filling and object removal by exemplar-based image inpainting." IEEE TIP (2004).

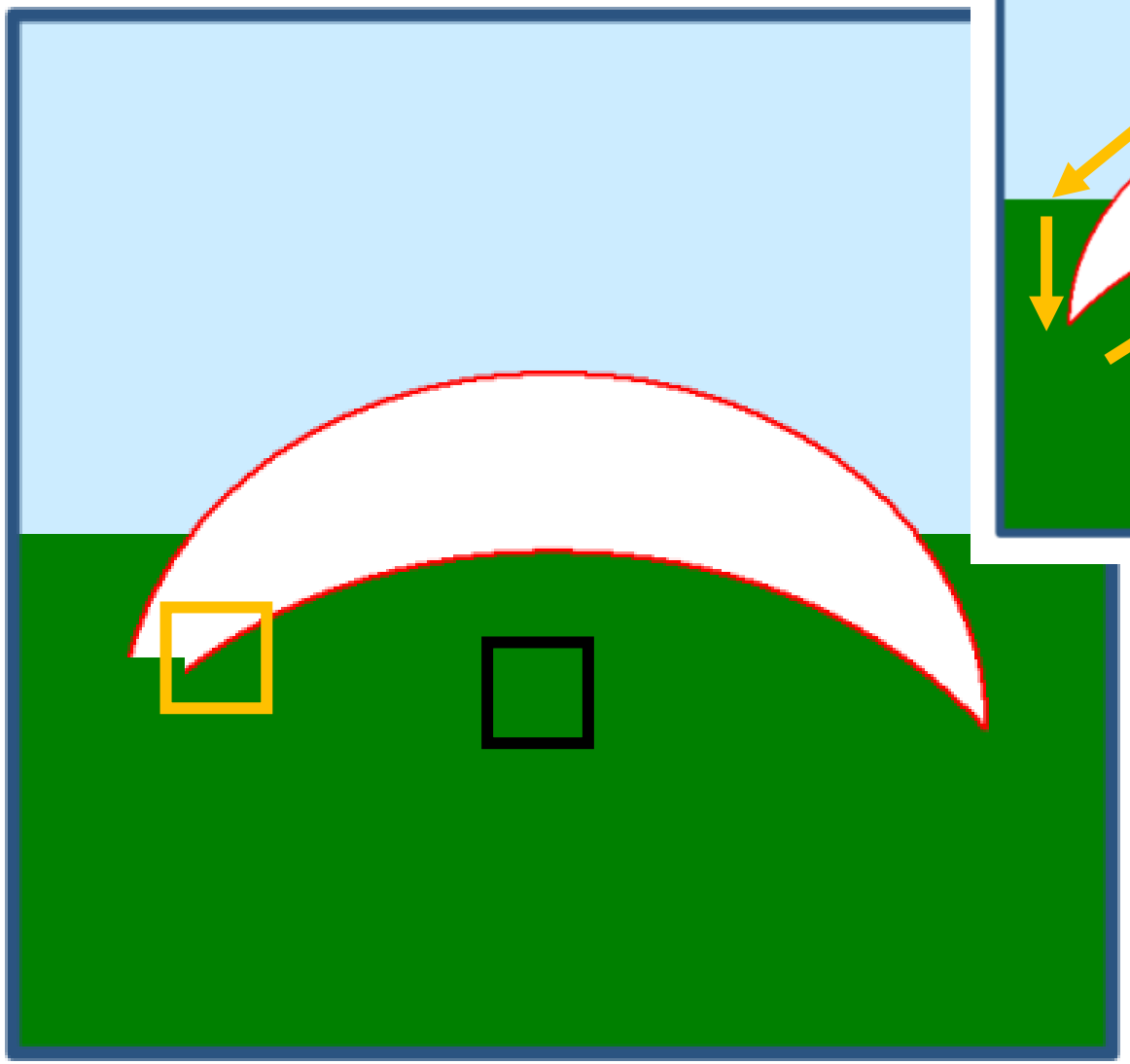
Example

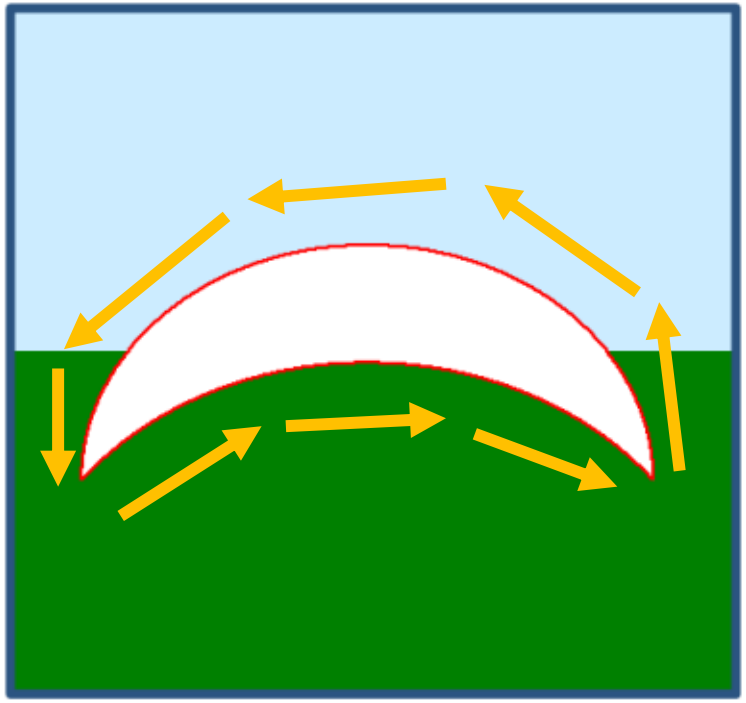


Example



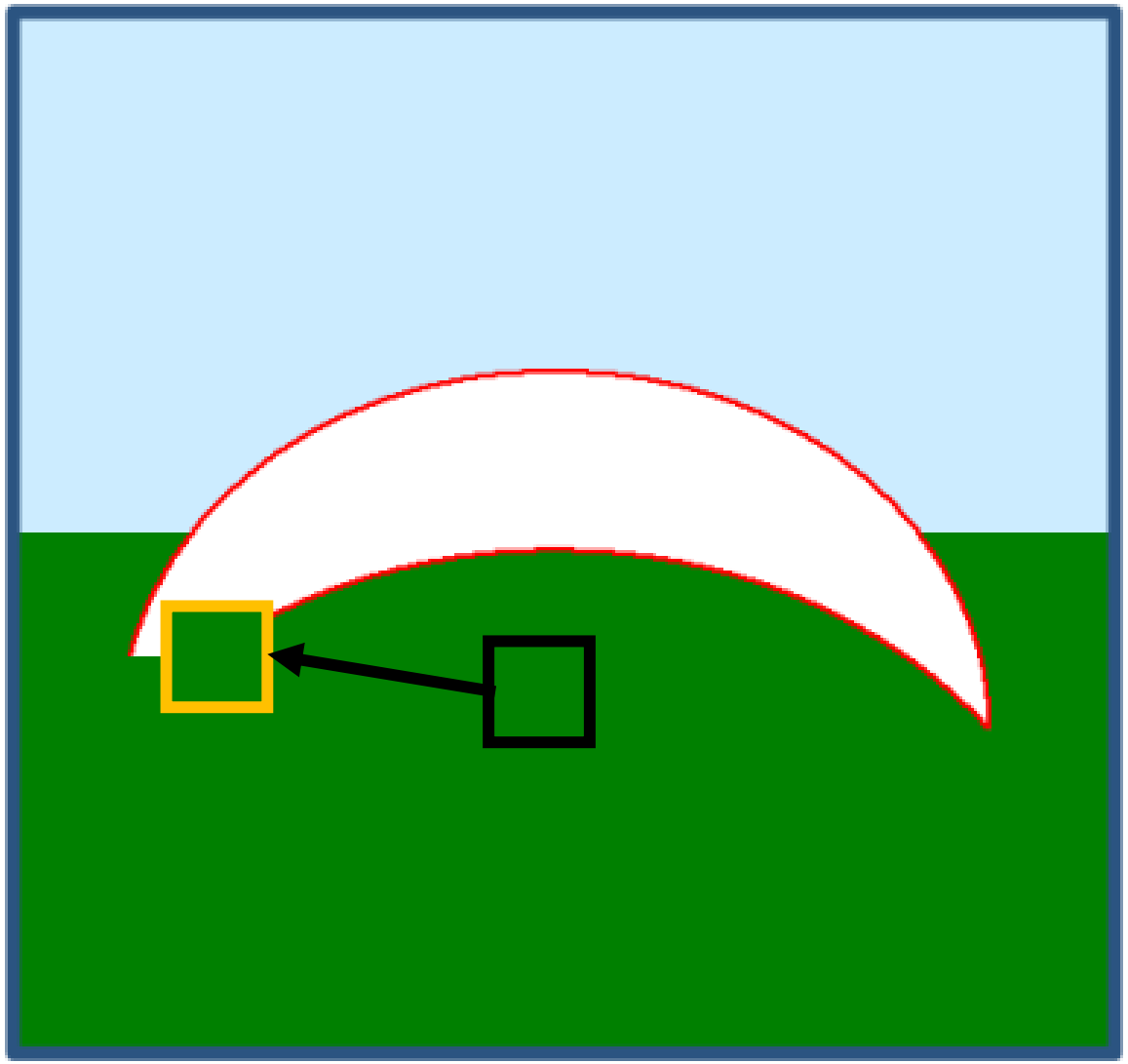
Example



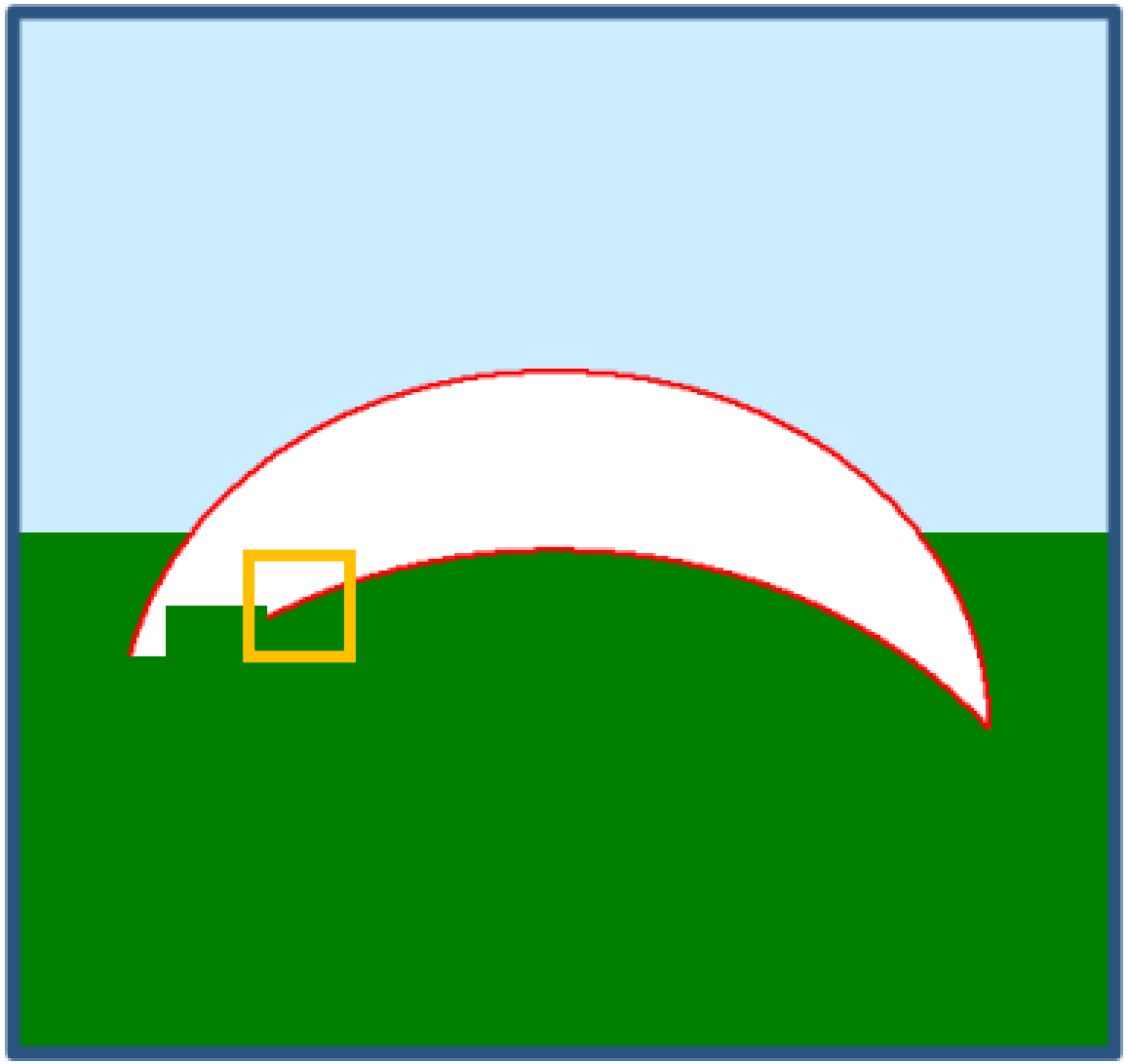


Onion-peeling

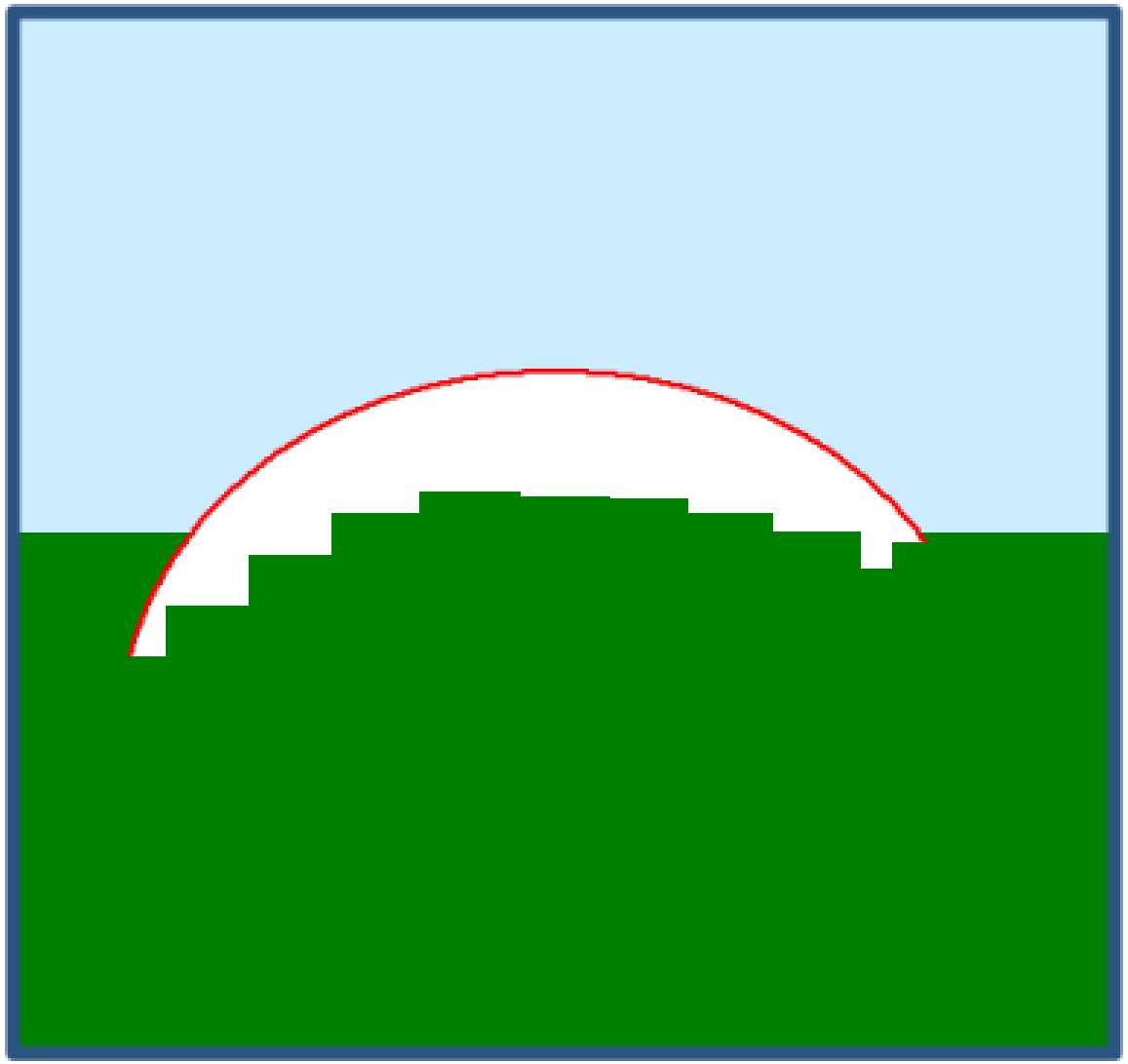
Example



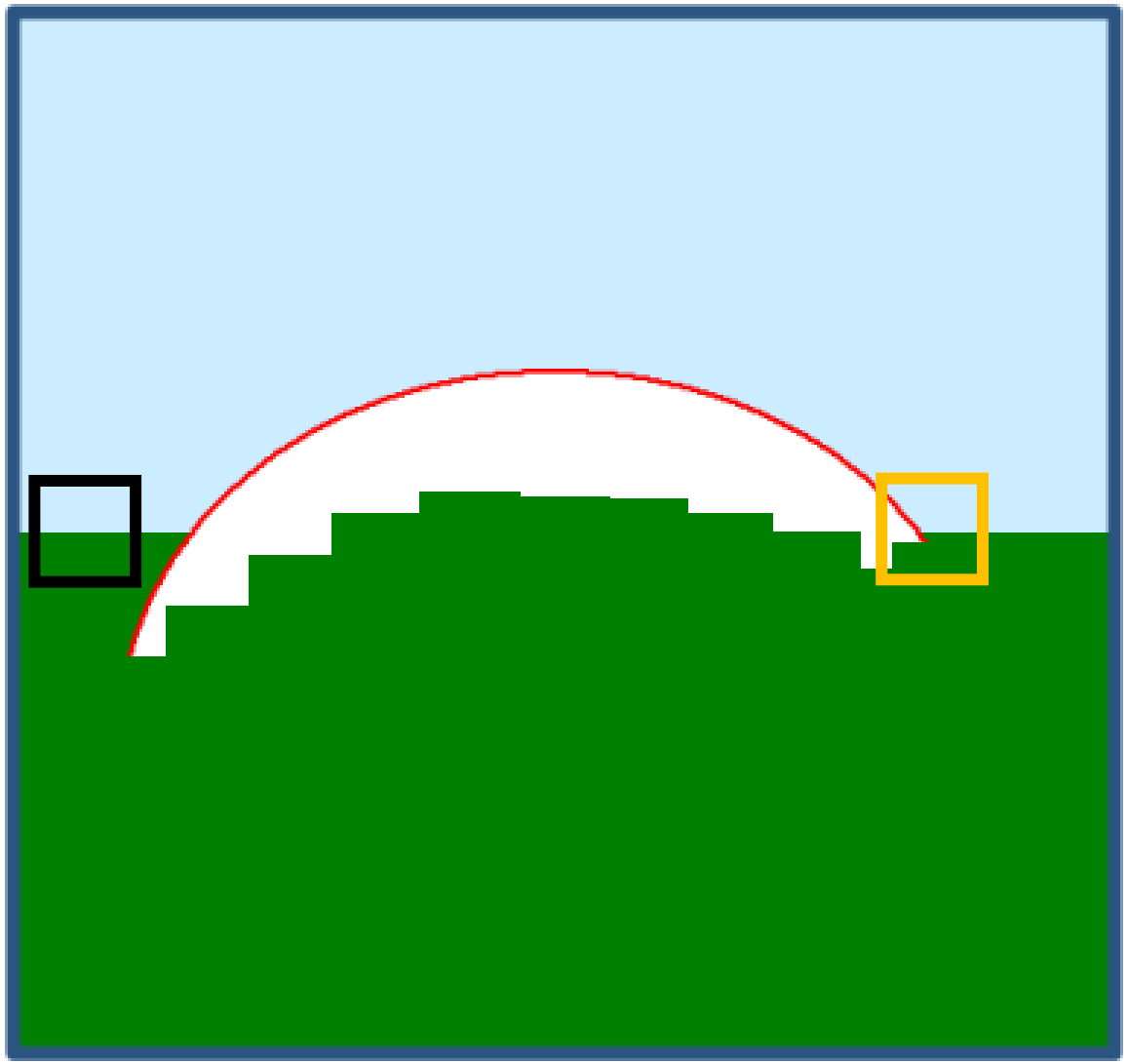
Example



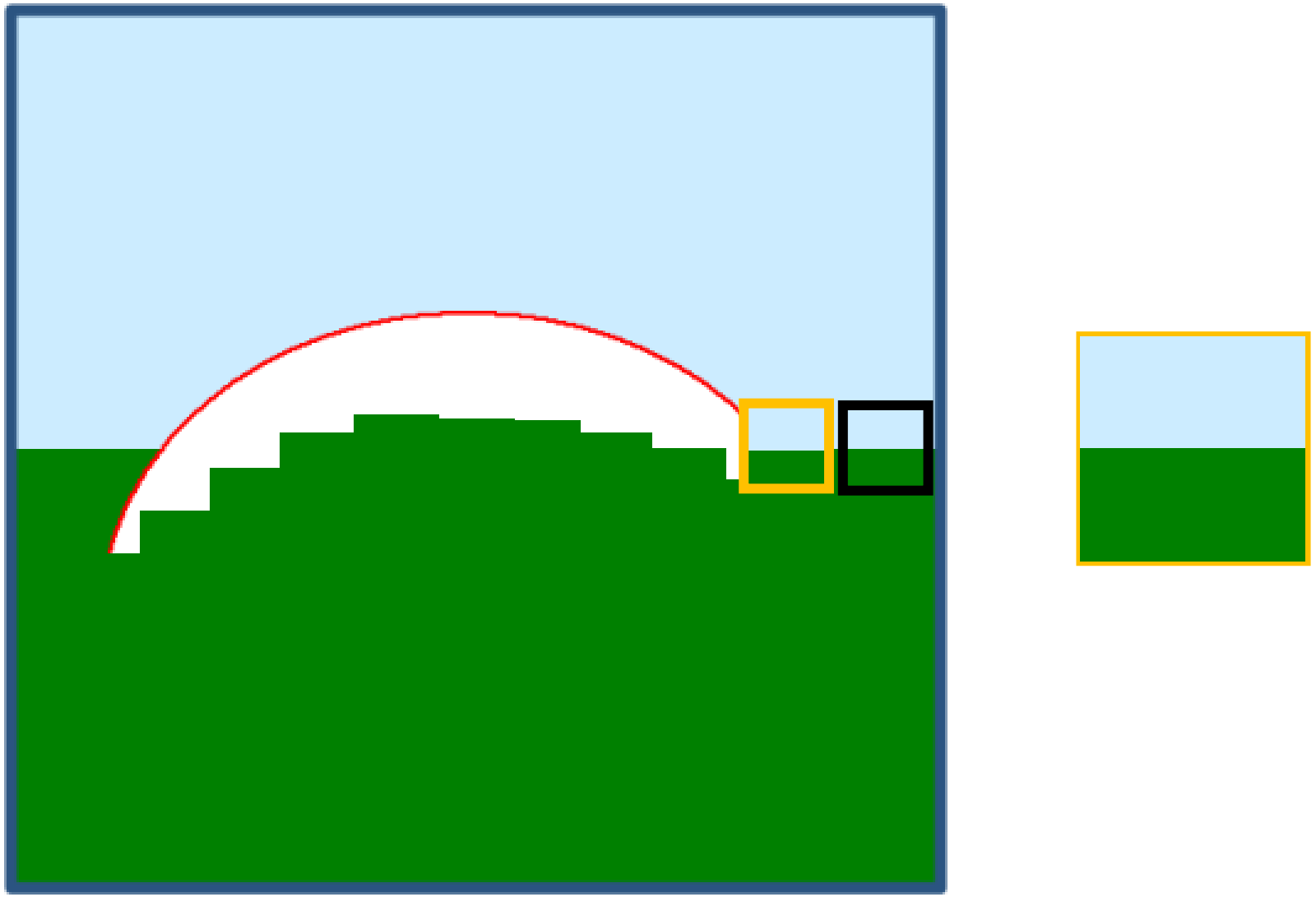
Example



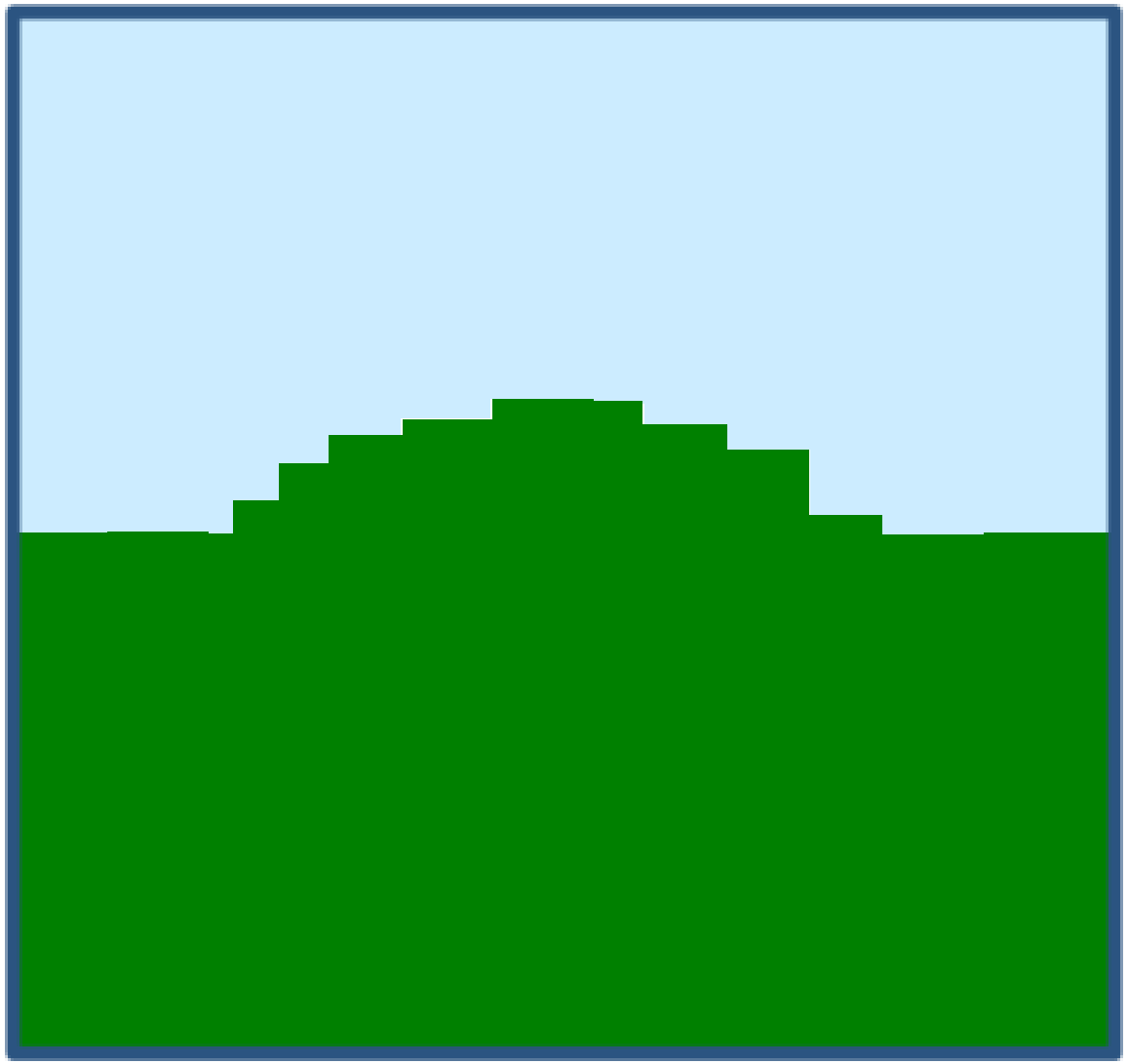
Example



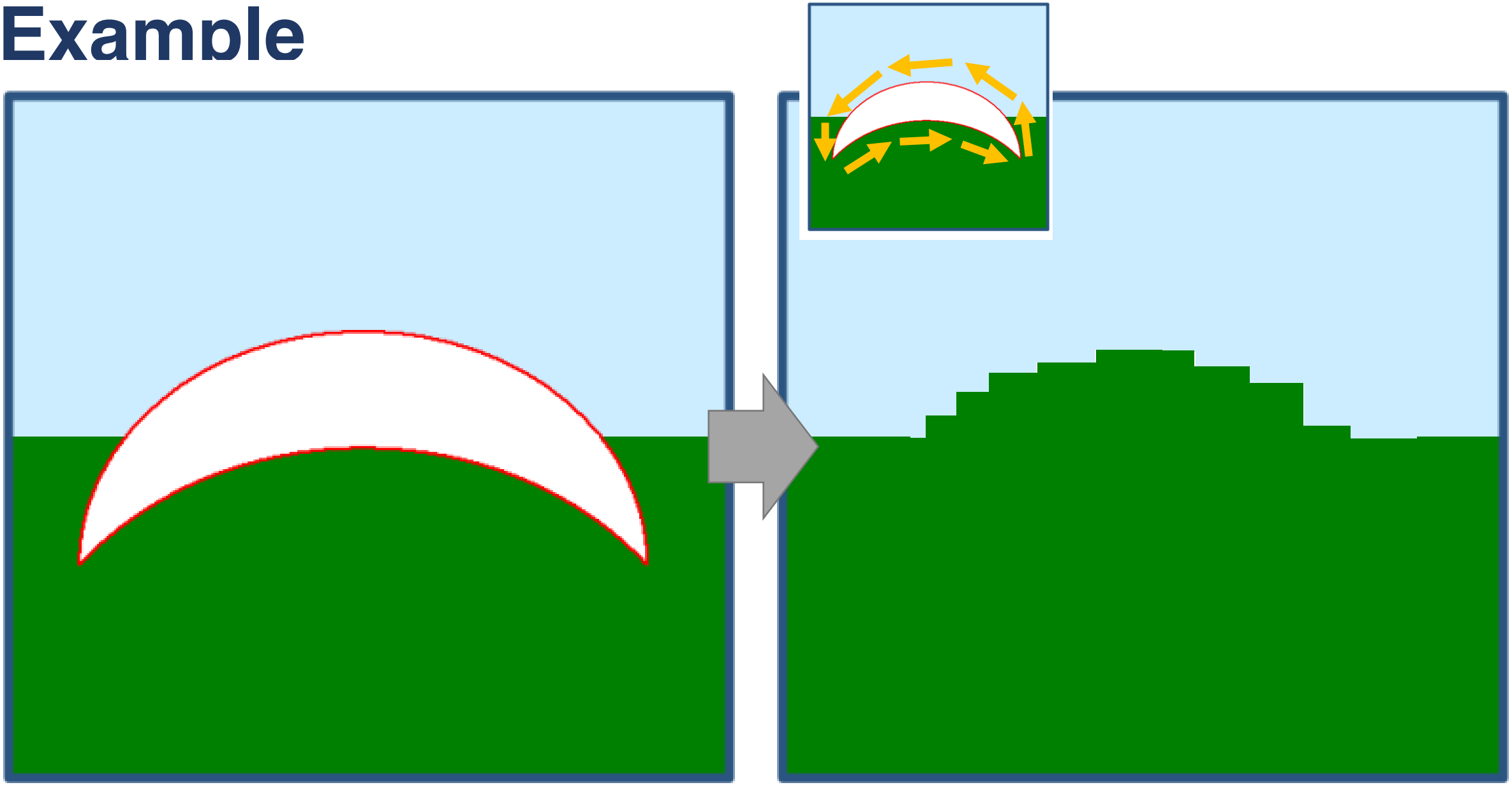
Example



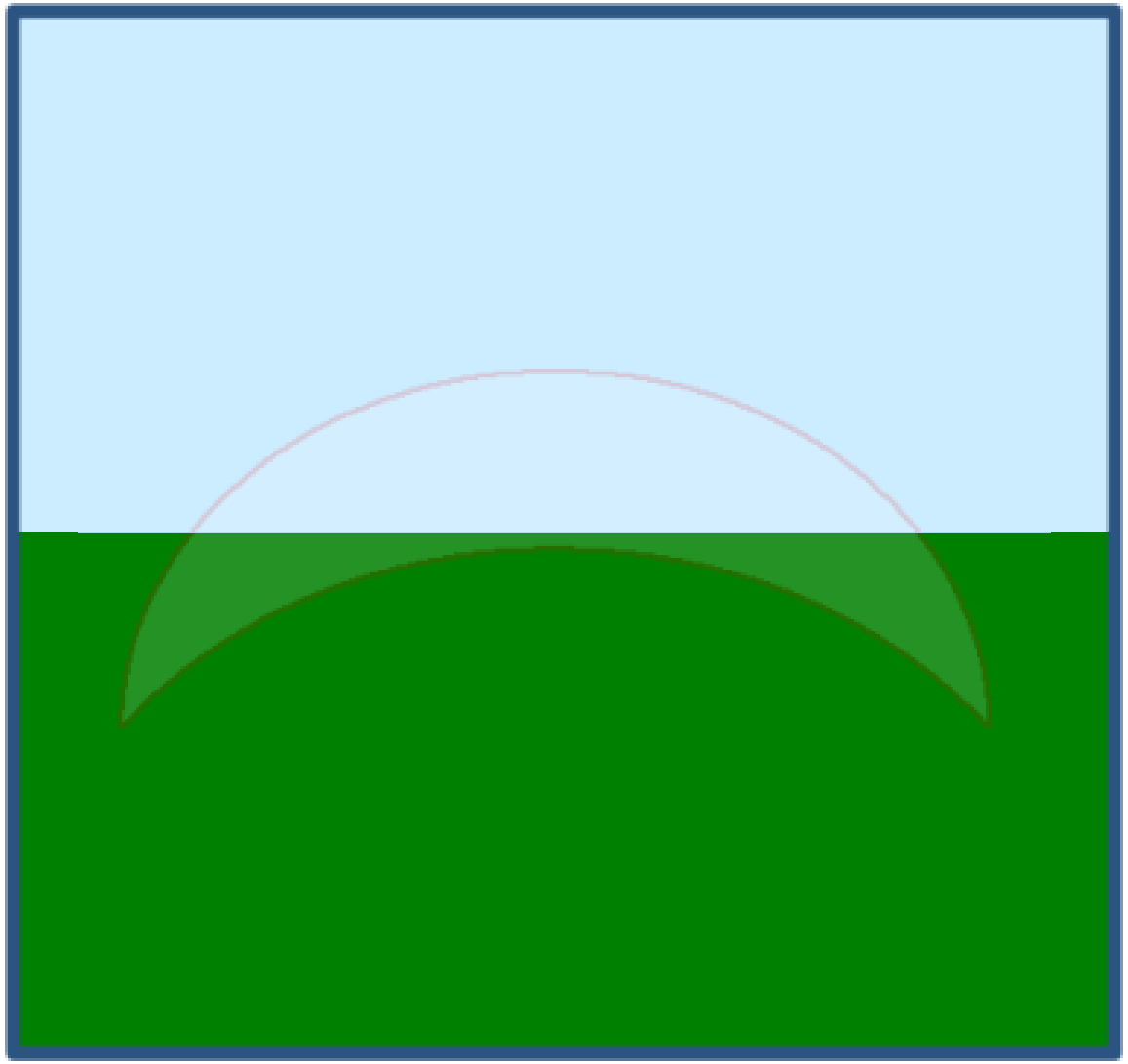
Example

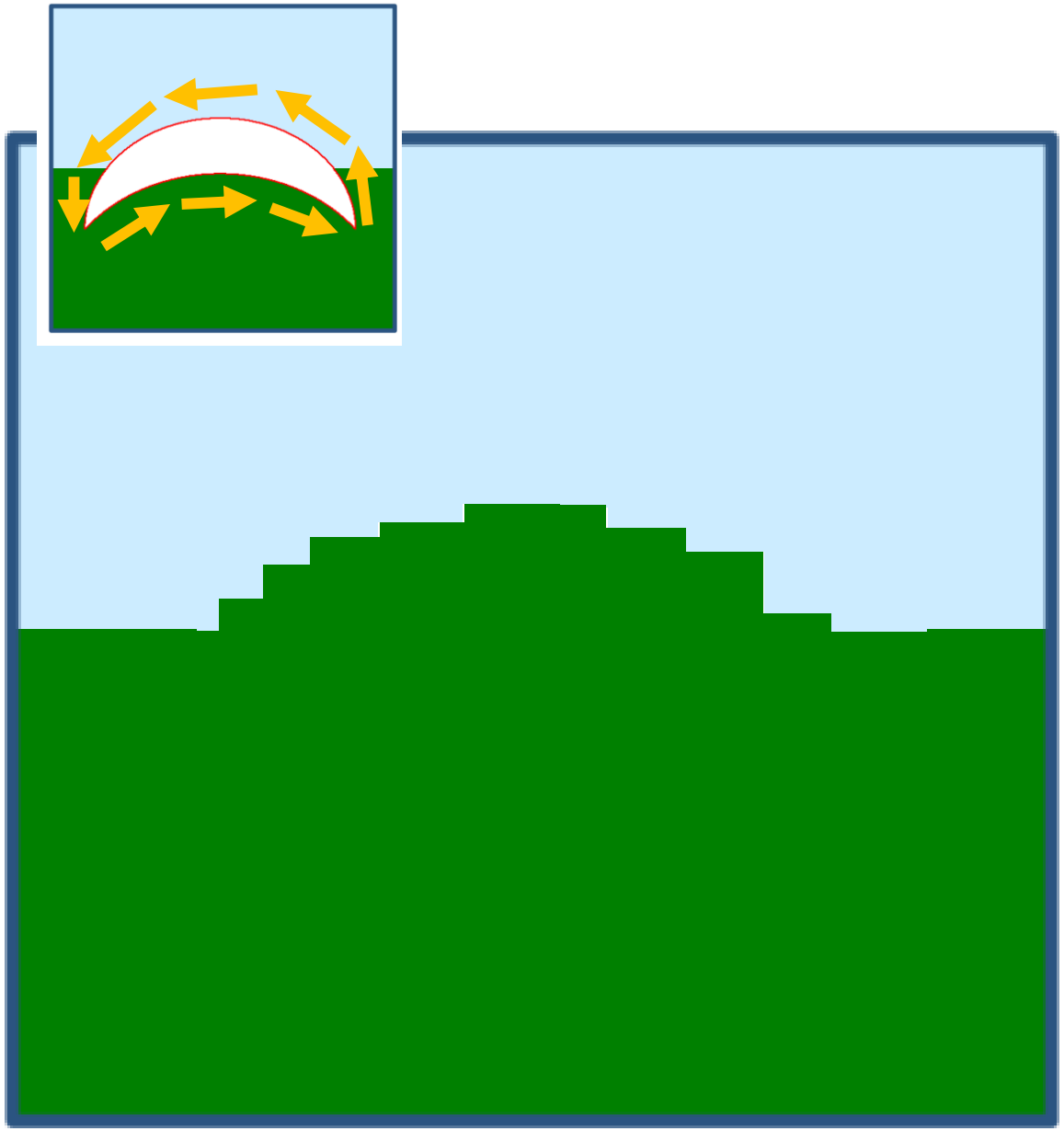


Example

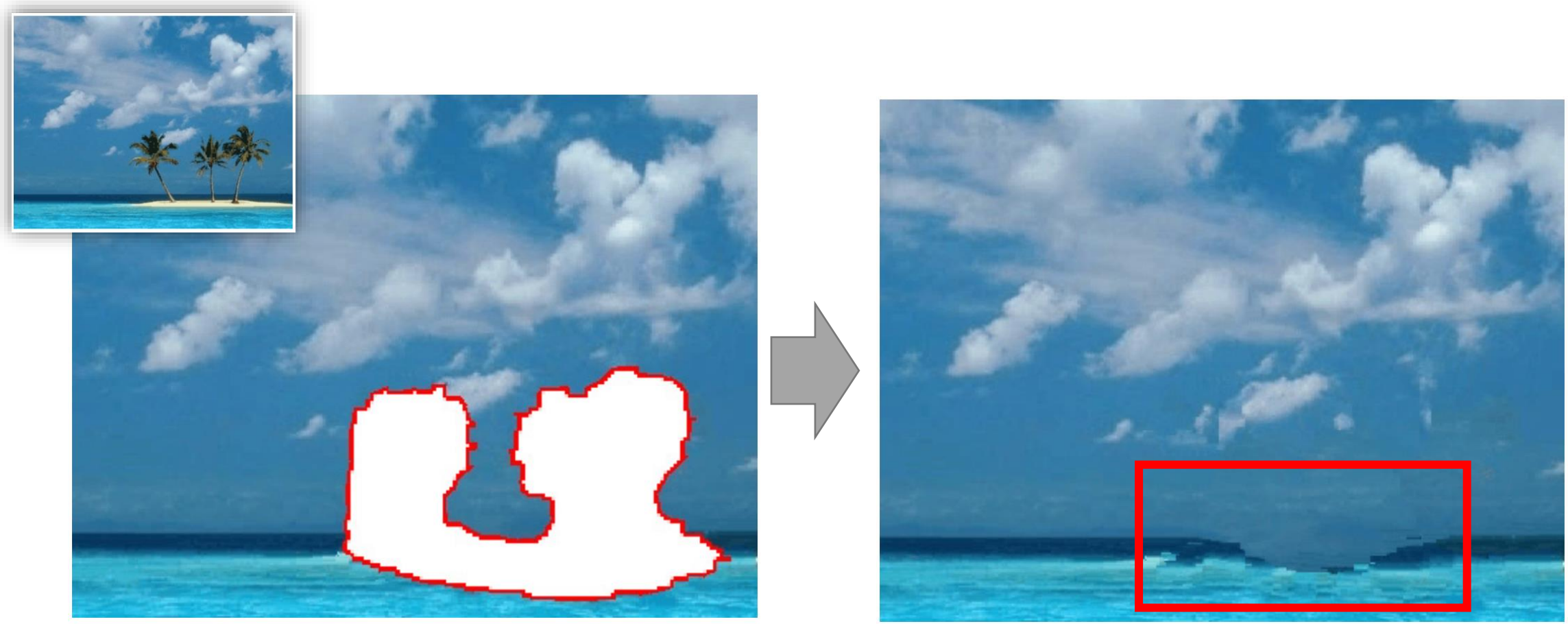


Example





Real case

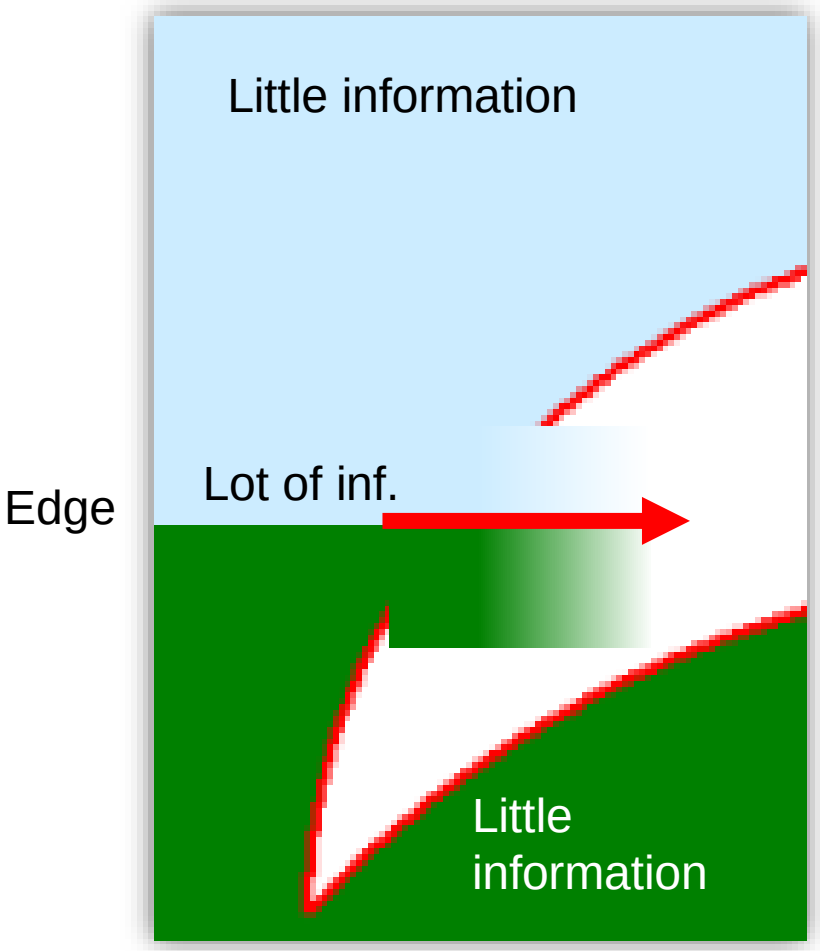


Criminisi, Antonio, Patrick Pérez, and Kentaro Toyama. "Region filling and object removal by exemplar-based image inpainting." IEEE TIP (2004).

Patch ordering

- We are more **confident** inserting pieces **where** we have a lot of **information**.
- **Edges** in the image **provide** lot of **information** about the content.
- **Edges** that are interrupted by the “hole” are more likely to **continue** than stop.

=> We start where strong edges head-on collide with the hole boundary.

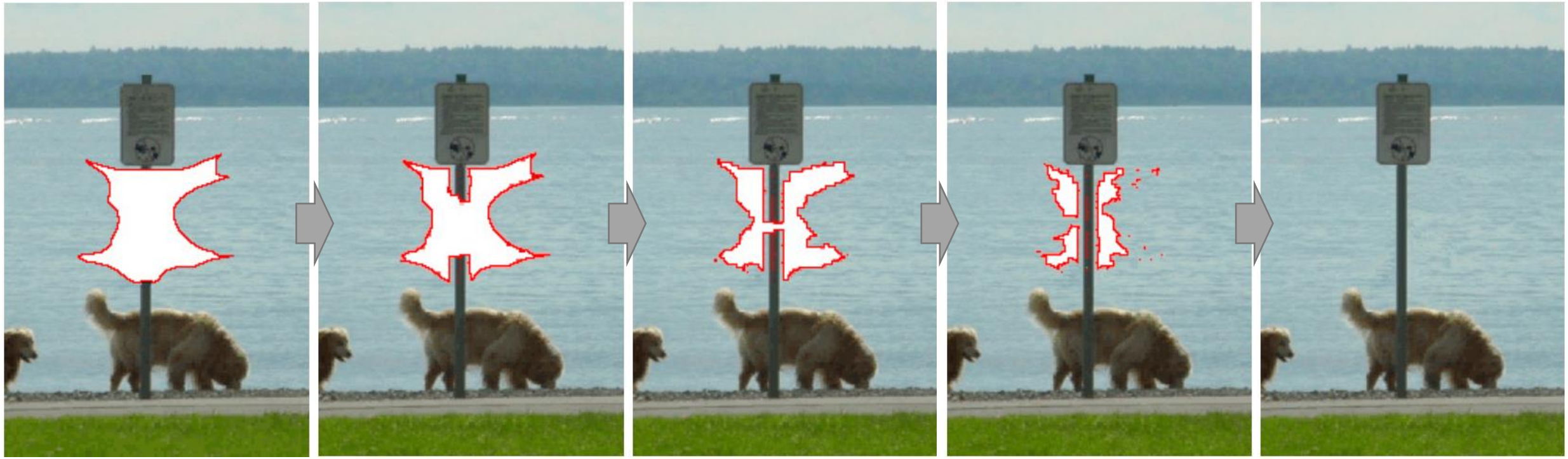


Example

“We start where **strong edges** head-on collide with the hole boundary.”

Criminisi, Antonio, Patrick Pérez, and Kentaro Toyama. "Region filling and object removal by exemplar-based image inpainting." IEEE TIP (2004).

Result



Step #0 Step #1 Step #2 Step #3 Step #N

Criminisi, Antonio, Patrick Pérez, and Kentaro Toyama. "Region filling and object removal by exemplar-based image inpainting." IEEE TIP (2004).

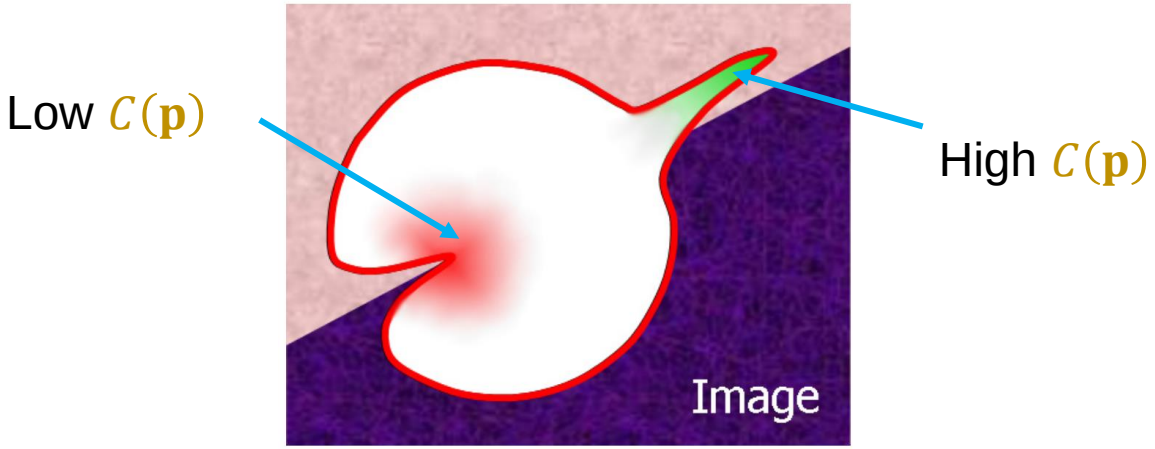
Patch priority

(select the patch with max priority)

$$P(\mathbf{p}) = C(\mathbf{p}) \cdot D(\mathbf{p})$$

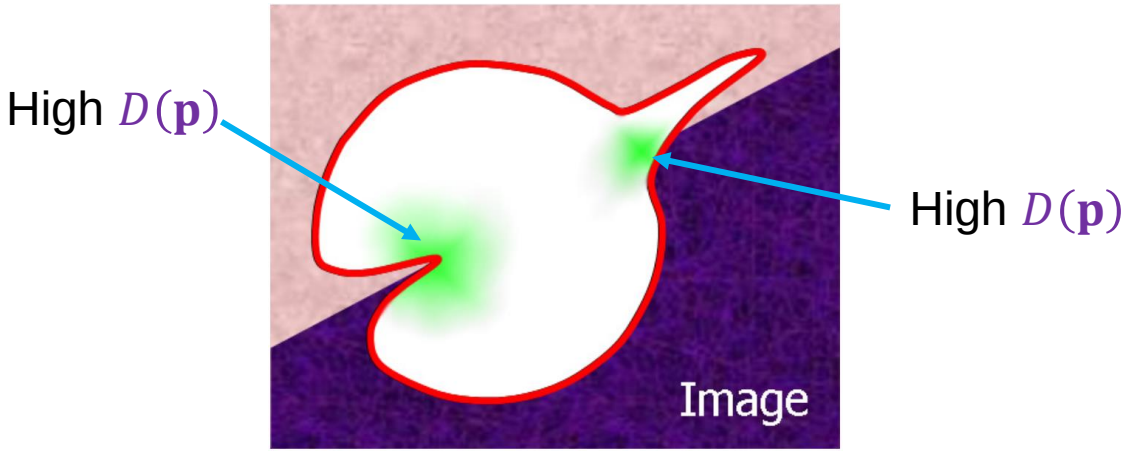
1. Confidence term $C(\mathbf{p})$

Prefer patches with only **a few missing pixels**.



2. Data term $D(\mathbf{p})$

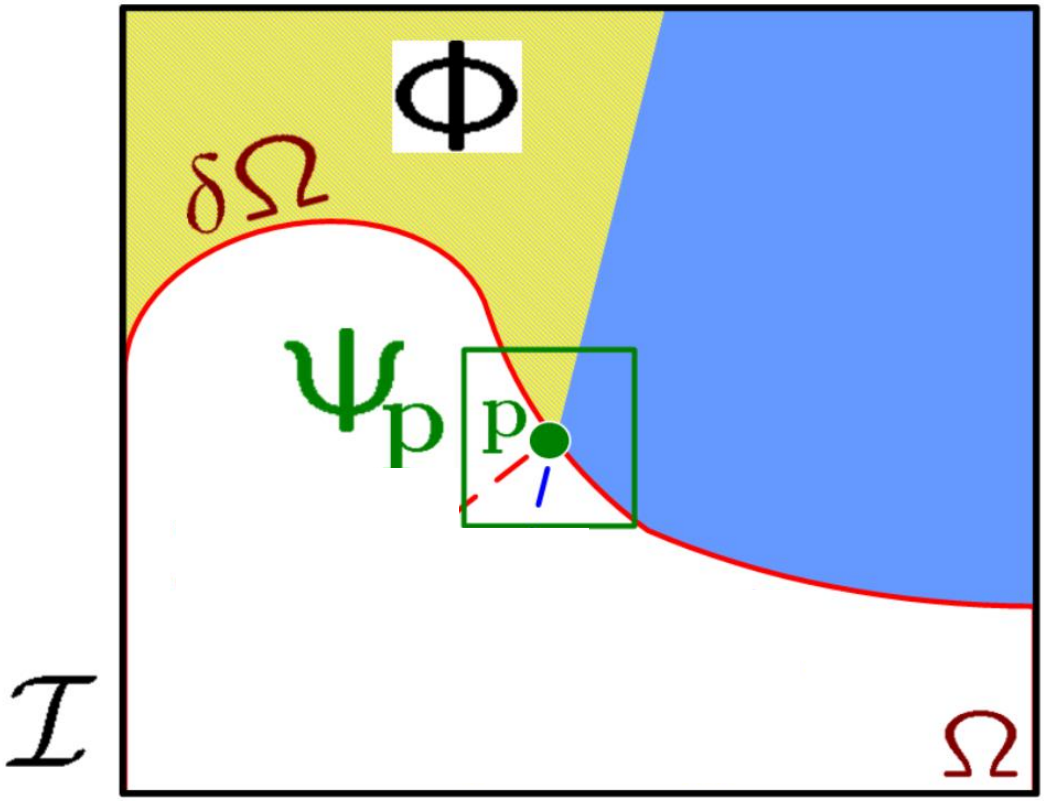
Prefer patches where **strong edges** head-on collide with the hole boundary.



Criminisi, Antonio, Patrick Pérez, and Kentaro Toyama. "Region filling and object removal by exemplar-based image inpainting." IEEE TIP (2004).

1. Confidence term

$$C_k(\mathbf{p}) = \frac{\sum_{\mathbf{q}\in\Psi_\mathbf{p}\cap(\mathcal{I}-\Omega)} C_{k-1}(\mathbf{q})}{|\Psi_\mathbf{p}|}$$



Notation

- $\mathbf{p}$ … a pixel of image $\mathcal{I}$
- Ω … set of missing pixels ($\delta\Omega$ = its border)
- $\Psi_\mathbf{p}$ … neighborhood of $\mathbf{p}$ (= patch)
- $|\Psi_\mathbf{p}|$ … number of (all) pixels in the patch
- $\Psi_\mathbf{p} \cap (\mathcal{I} - \Omega)$ … valid pixels in the patch

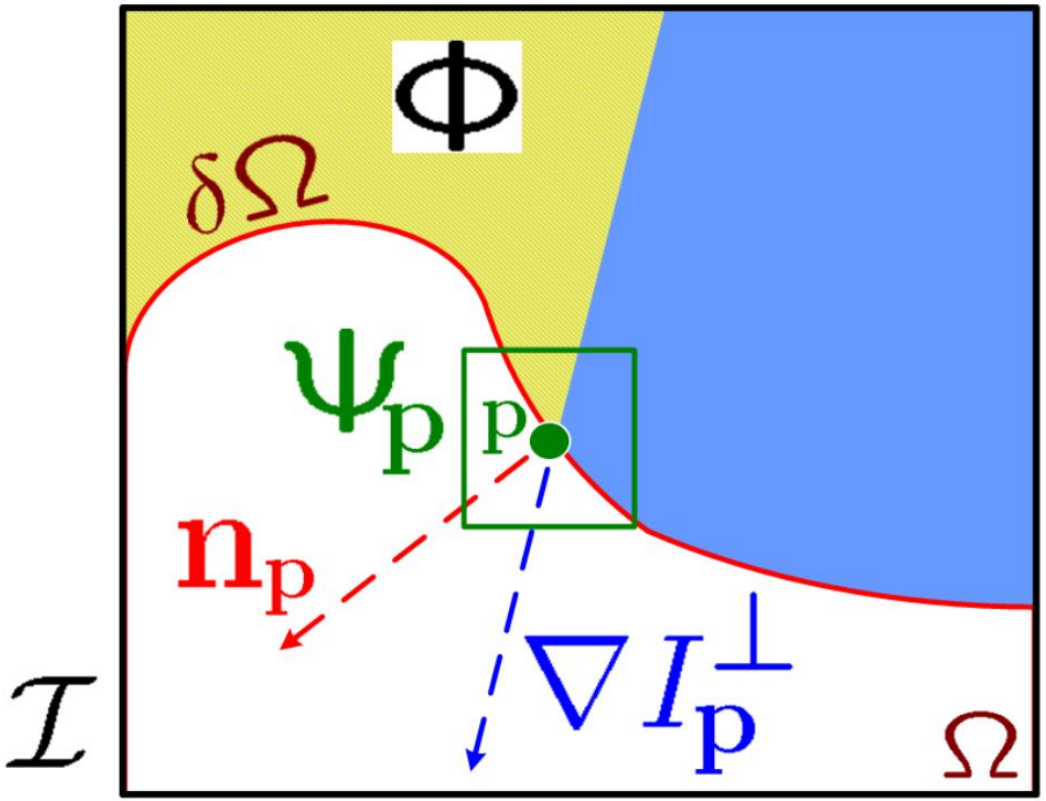
Initialization

- $C_0(\mathbf{p}) = 0, \forall \mathbf{p} \in \Omega$
- $C_0(\mathbf{p}) = 1, \forall \mathbf{p} \in \mathcal{I} - \Omega$

Criminisi, Antonio, Patrick Pérez, and Kentaro Toyama. "Region filling and object removal by exemplar-based image inpainting." IEEE TIP (2004).

2. Data term

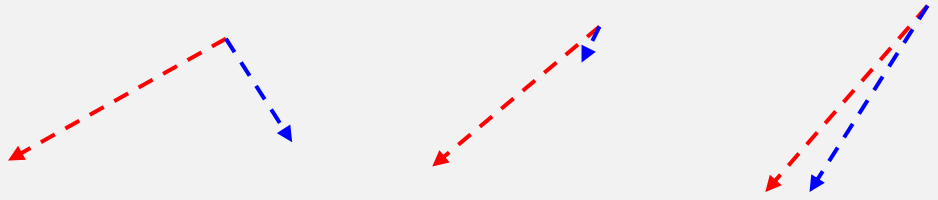
$$D_k(\mathbf{p}) = \left| \nabla I_{\mathrm{p}}^{\perp} \cdot \mathbf{n}_{\mathrm{p}} \right|$$



Notation

- ∇I_{p}… image gradient at pixel $\mathbf{p}$
- $\nabla I_{\mathrm{p}}^{\perp}$… isophote (direction and intensity)
- $\mathbf{n}_{\mathrm{p}}$ … normal of border $\delta\Omega$

Reminder: Properties of a **dot product**



$\left\|\nabla I_{\mathrm{p}}^{\perp} \cdot \mathbf{n}_{\mathrm{p}}\right\|$ is small	$\left\|\nabla I_{\mathrm{p}}^{\perp} \cdot \mathbf{n}_{\mathrm{p}}\right\|$ is small	$\left\|\nabla I_{\mathrm{p}}^{\perp} \cdot \mathbf{n}_{\mathrm{p}}\right\|$ is large

Criminisi, Antonio, Patrick Pérez, and Kentaro Toyama. "Region filling and object removal by exemplar-based image inpainting." IEEE TIP (2004).

Where to copy from?

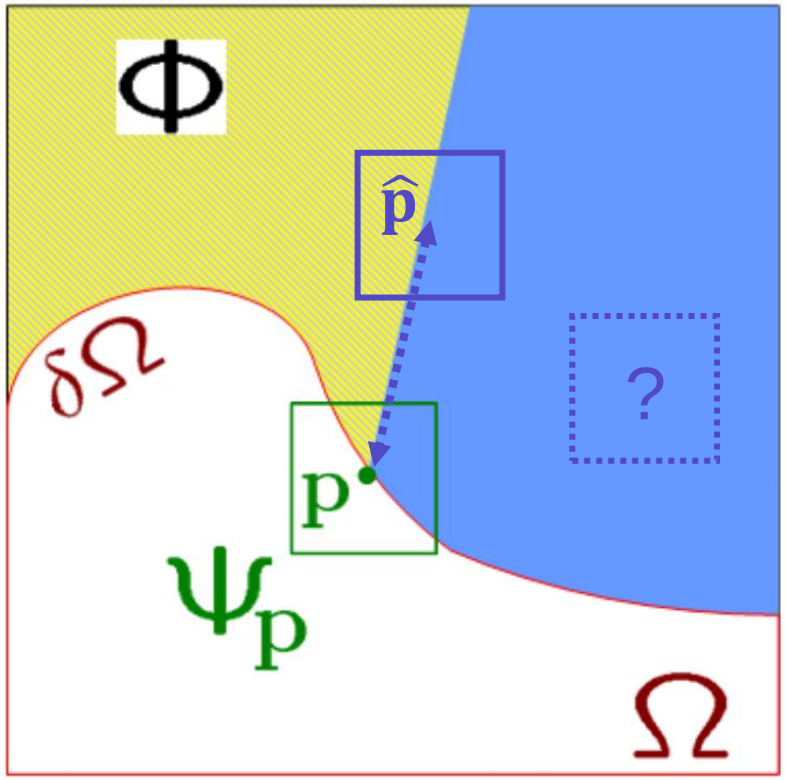
Find a patch with the minimal

Sum of Squared Differences (SSD)

$$SSD(\mathbf{p}, \hat{\mathbf{p}}) = \sum_i (\mathrm{p}_i - \hat{\mathrm{p}}_i)^2$$

$$\left(\begin{bmatrix} 1 & 7 & 2 \\ 3 & 5 & 2 \\ 3 & 6 & 3 \end{bmatrix} - \begin{bmatrix} 0 & 6 & 2 \\ 4 & 0 & 2 \\ 6 & 5 & 2 \end{bmatrix} \right)^2 = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 25 & 0 \\ 9 & 1 & 1 \end{bmatrix} \quad \text{SSD=39}$$

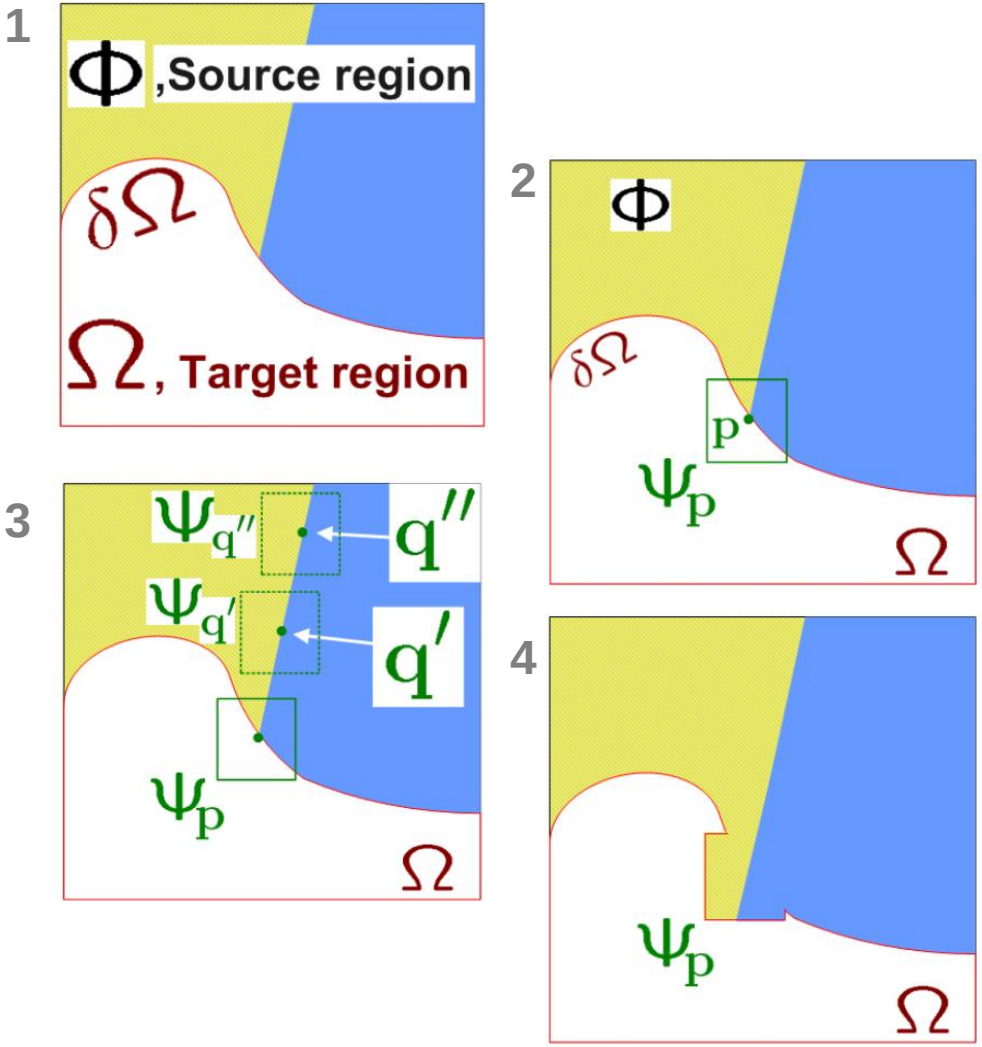
Note: **We skip the invalid pixels!**



Criminisi, Antonio, Patrick Pérez, and Kentaro Toyama. "Region filling and object removal by exemplar-based image inpainting." IEEE TIP (2004).

Algorithm summary

1. Compute pixel priorities.
2. Select a patch with the top priority.
3. Find the most similar source patch.
4. Copy the patch over.

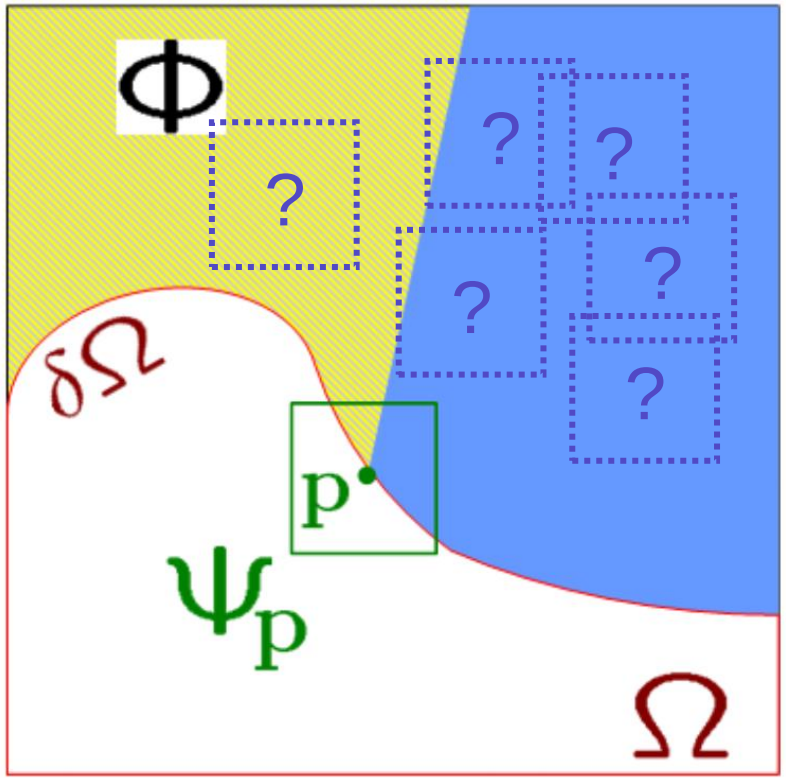


Criminisi, Antonio, Patrick Pérez, and Kentaro Toyama. "Region filling and object removal by exemplar-based image inpainting." IEEE TIP (2004).

Searching for matches is expensive

- **Situation**: There are as many patches as there are pixels.

- **Naïve solution**: Exhaustive search.
  - Very costly.



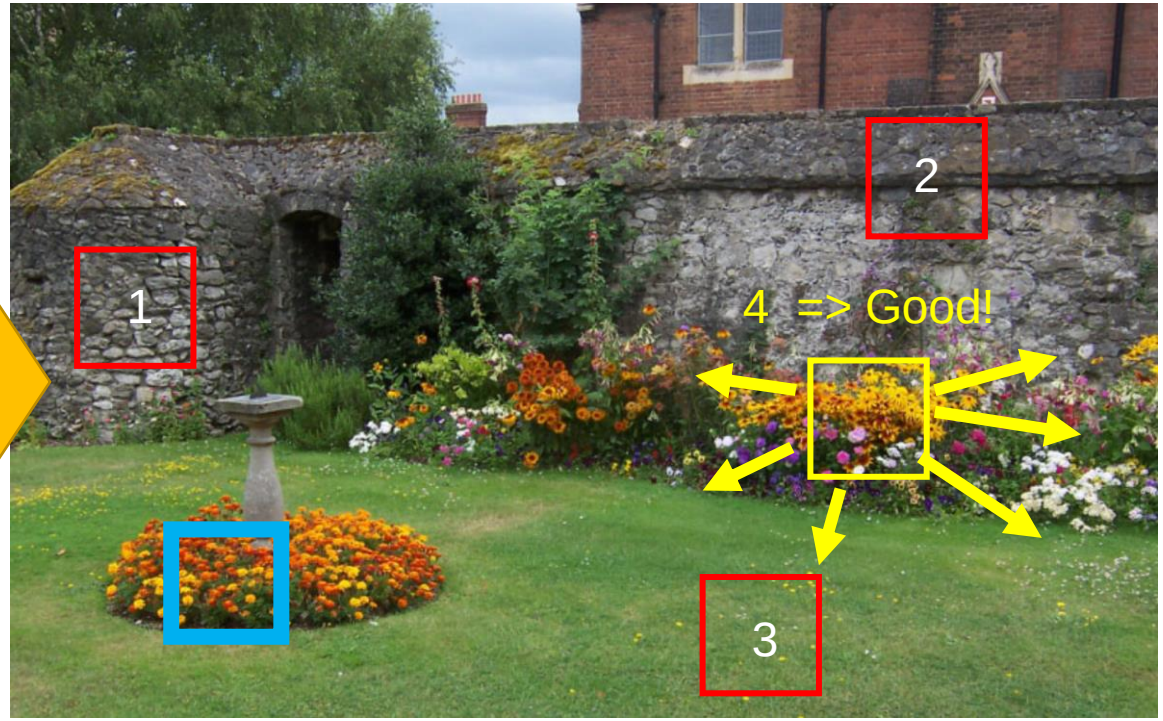
Is there a better way?

Criminisi, Antonio, Patrick Pérez, and Kentaro Toyama. "Region filling and object removal by exemplar-based image inpainting." IEEE TIP (2004).

Patch Match

1. Make random guesses.
2. Propagate/improve good guesses.
3. Repeat.

Exhaustive search



Patch Match

Barnes, Connelly, et al. "PatchMatch: A randomized correspondence algorithm for structural image editing." ACM SIGGRAPH (2009)

Example

Input image

Inpainted image

Criminisi, Antonio, Patrick Pérez, and Kentaro Toyama. "Region filling and object removal by exemplar-based image inpainting." IEEE TIP (2004).

Limitations

- Can only inpaint what is present in the rest of the image.
- Small patches may not capture larger structures.

(Examples using Patch Match method.)

Input image

Inpainted image

James Hays and Alexei A. Efros: Scene Completion Using Millions of Photographs. SIGGRAPH 2007.

Exemplar-based inpainting

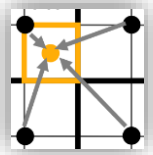
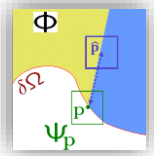
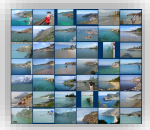
Good

- Preserves texture.
- Relatively fast.

Bad

- Limited to information in single image.
- Does not preserve global structure.

Algorithms

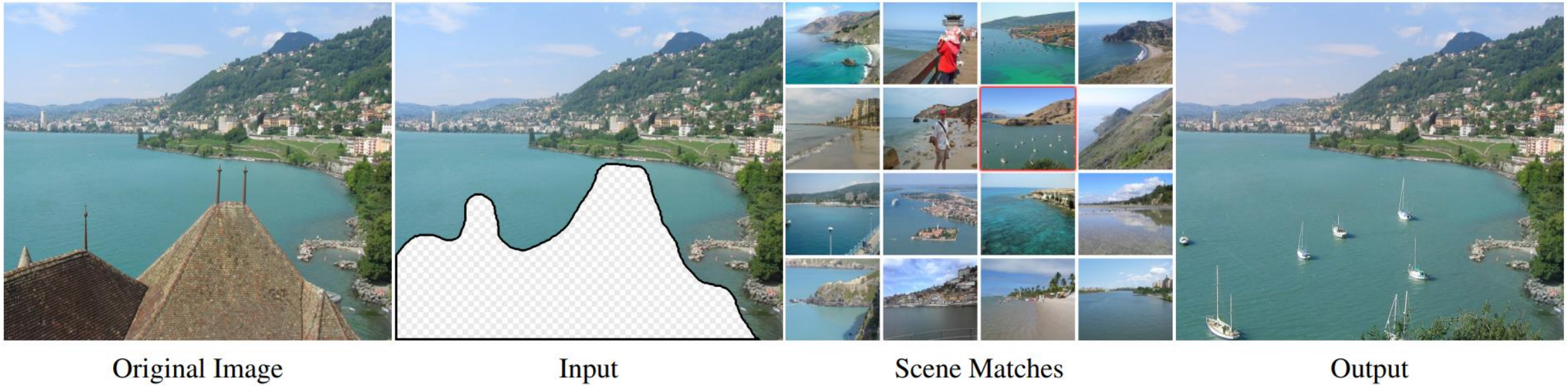
	Statistic		Prior
1. Pull-Push	Simple: Mean color		Single image
2. PDE inpainting	Simple: Gradients		Single image
3. Exemplar inpainting	Medium: Patterns		Single image
4. Collection sampling	Medium: Patterns		Collection
5. Neural inpainting	Complex: Multi-scale		Collection

Algorithms

	Statistic		Prior
1. Pull-Push	Simple: Mean color		Single image
2. PDE inpainting	Simple: Gradients		Single image
3. Exemplar inpainting	Medium: Patterns		Single image
4. Collection sampling	Medium: Patterns		Collection
5. Neural inpainting	Complex: Multi-scale		Collection

Collection sampling

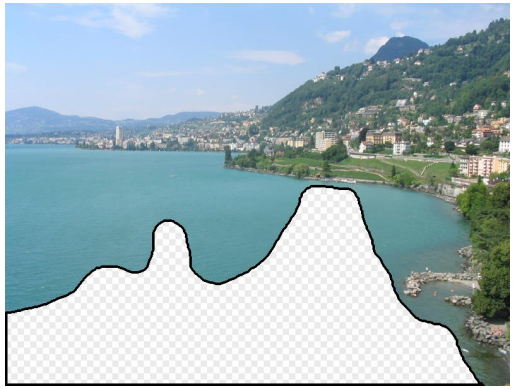
Idea: Copy a matching region from a similar image in a collection.



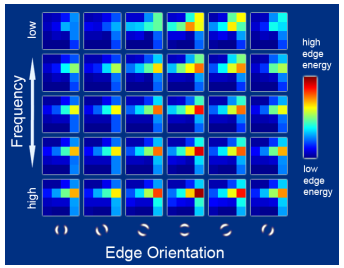
James Hays and Alexei A. Efros: Scene Completion Using Millions of Photographs. SIGGRAPH 2007.

Algorithm

1. Input image



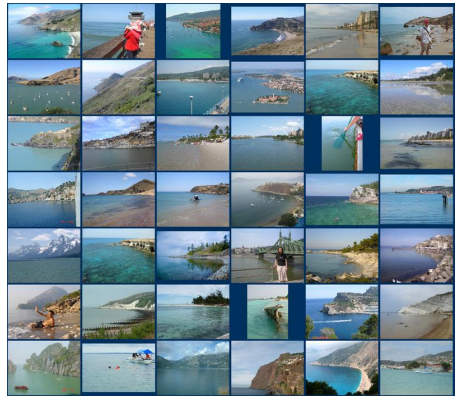
2. Compute descriptor



+

3. Image collection

4. Find k-nearest matches



5. Select and align

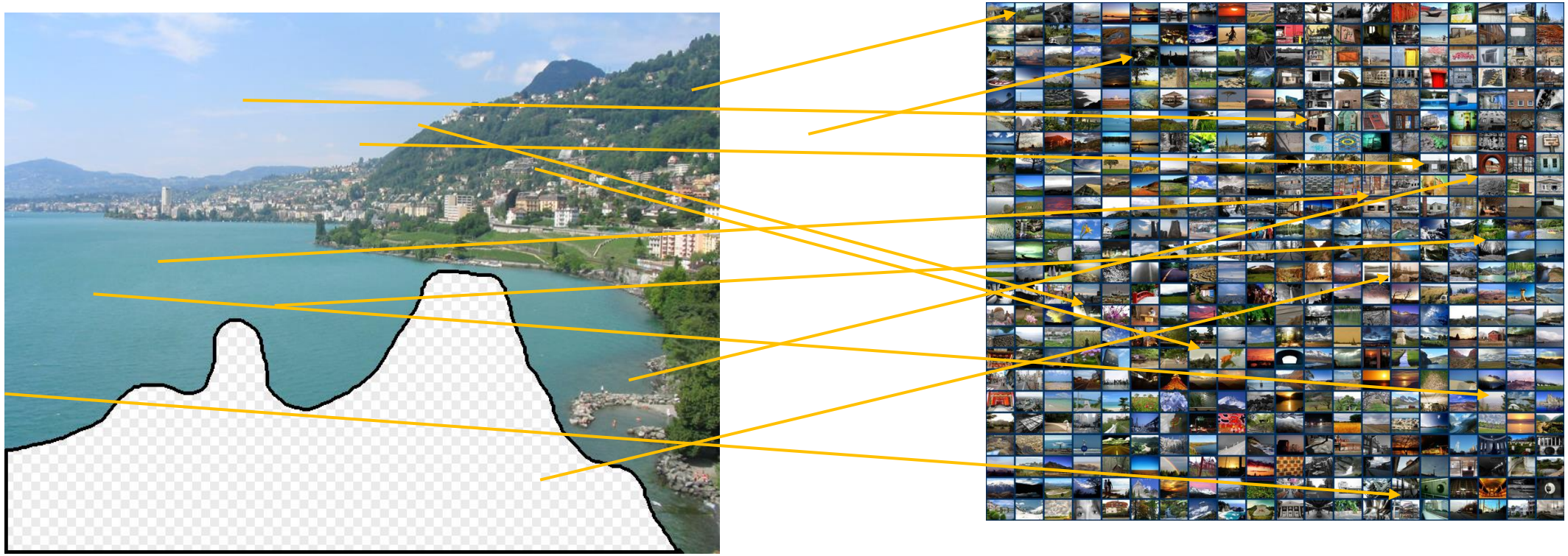
6. Merge

James Hays and Alexei A. Efros: Scene Completion Using Millions of Photographs. SIGGRAPH 2007.

Step 2: Image descriptor

We cannot match image to the collection pixel by pixel.

=> Find a compact descriptor.



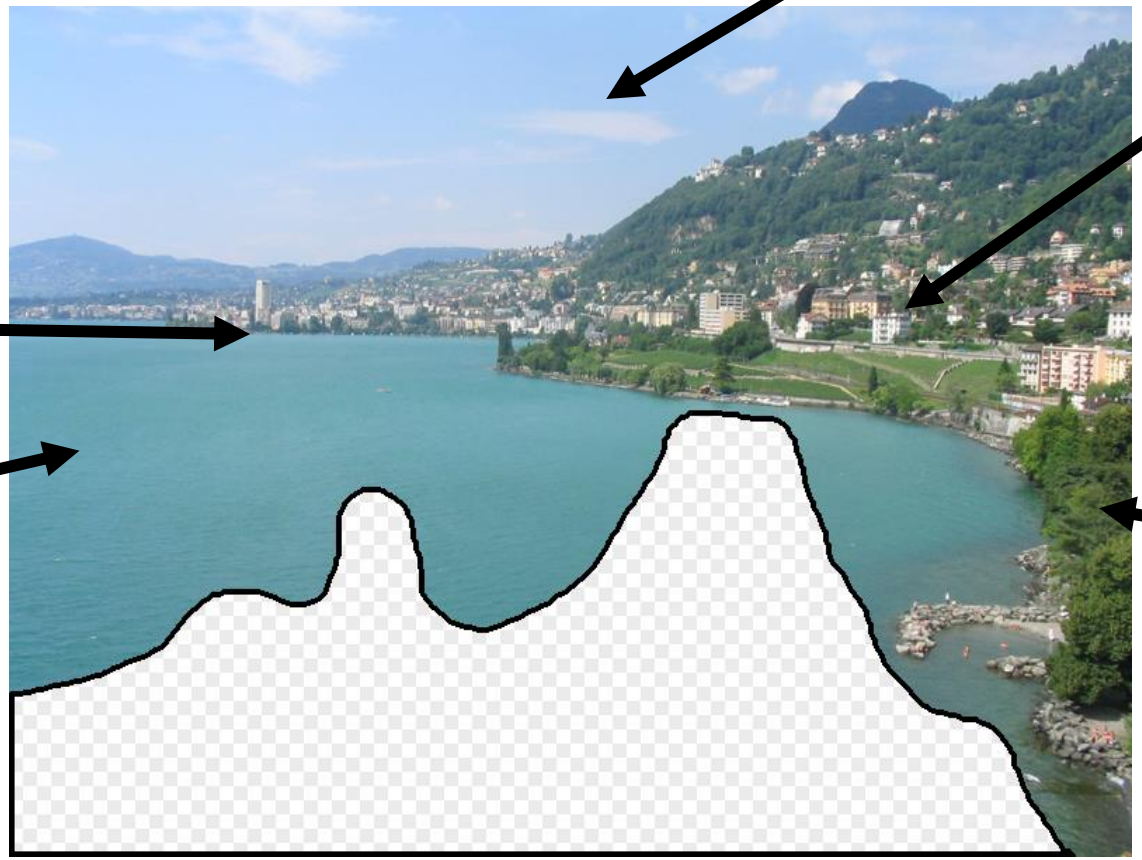
James Hays and Alexei A. Efros: Scene Completion Using Millions of Photographs. SIGGRAPH 2007.

GIST descriptor

Extract the “gist” (spirit) of the images.

Smooth edges of the clouds

Horizontal and vertical edges of the buildings



Sharp horizontal edge of the coastline

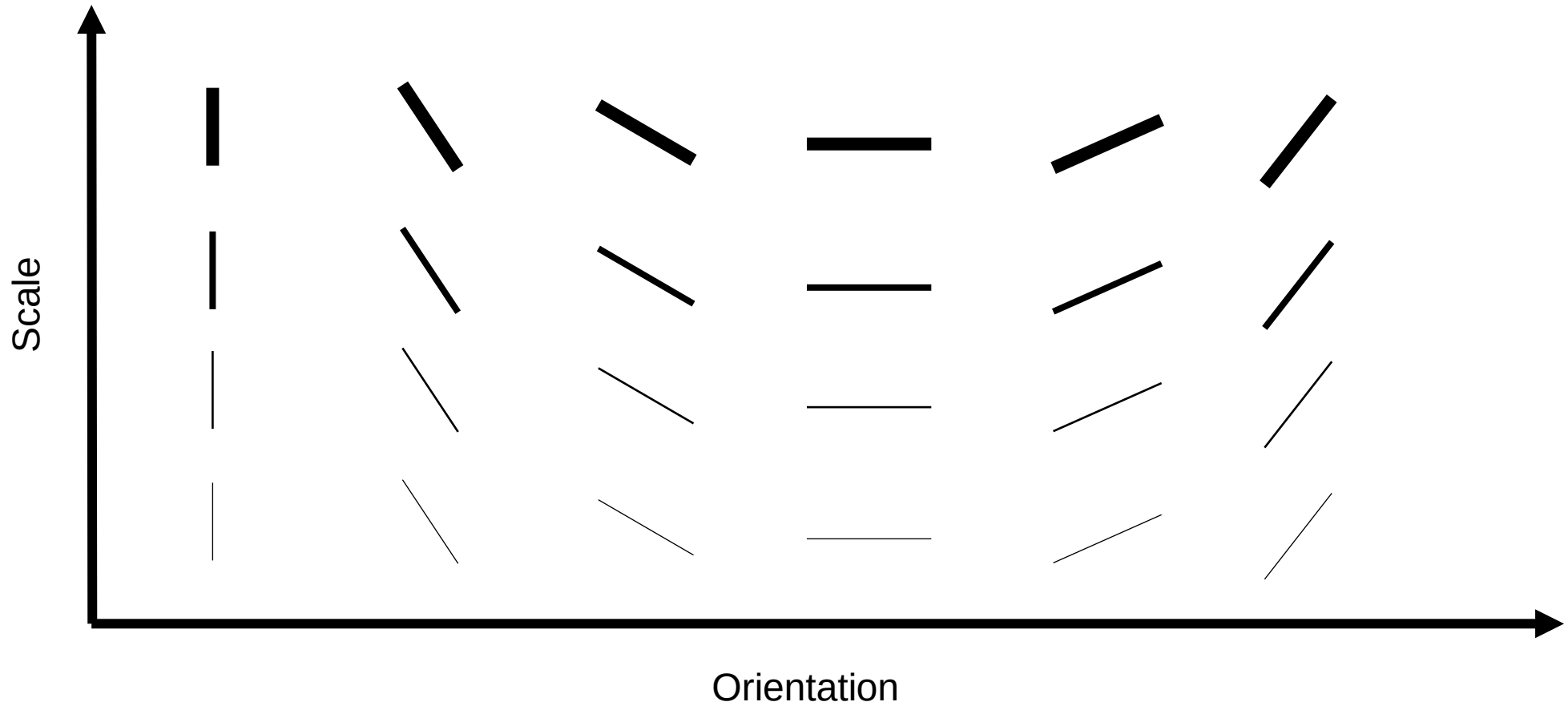
Almost no edges

Irregular edges of nature

Torralba, Antonio, et al. "Context-based vision system for place and object recognition." *Computer Vision, IEEE International Conference on*. Vol. 2. IEEE Computer Society, 2003.

Image decomposition

Describe edges at each scale and orientation separately.

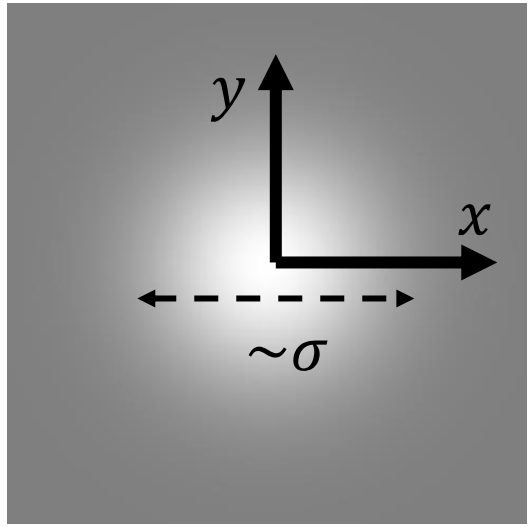


Gabor filter

Sensitive to edges of selected scale and orientation.

Gaussian

$$g(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

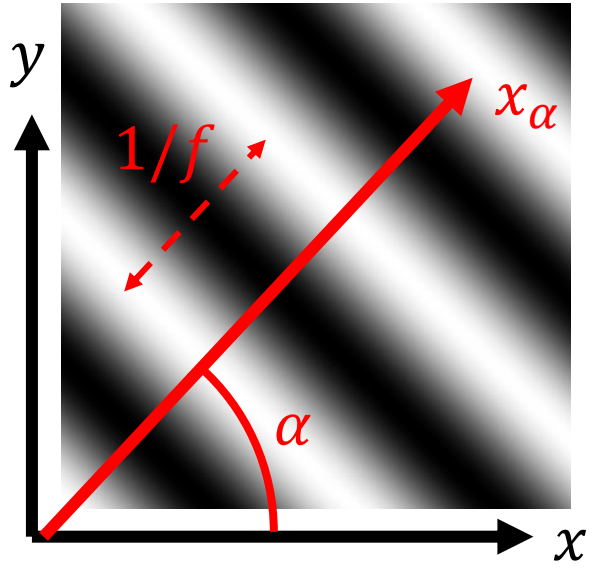


$\bullet$

Sinusoid (corrugation)

$$s(x,y) = \sin(x_\alpha f)$$

$$\mathrm{x}_\alpha(x,y) = \mathrm{x} \cdot \cos(\alpha) - y \cdot \sin(\alpha)$$

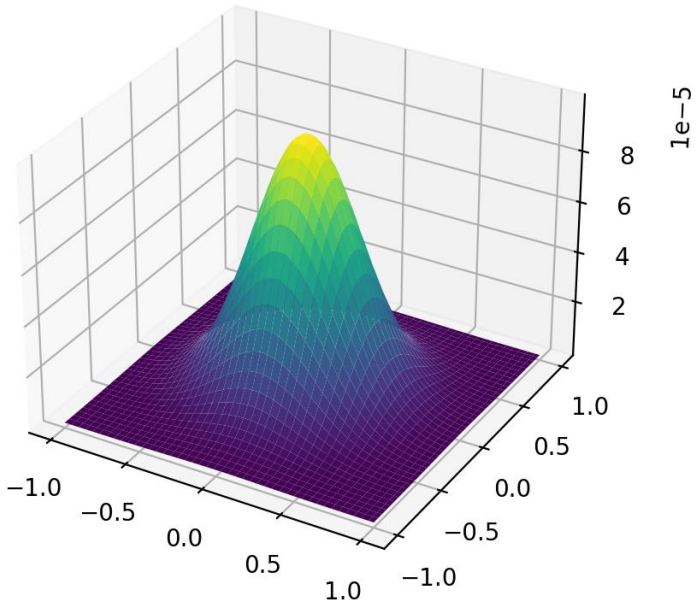


Gabor filter

Sensitive to edges of selected scale and orientation

Gaussian

$$g(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

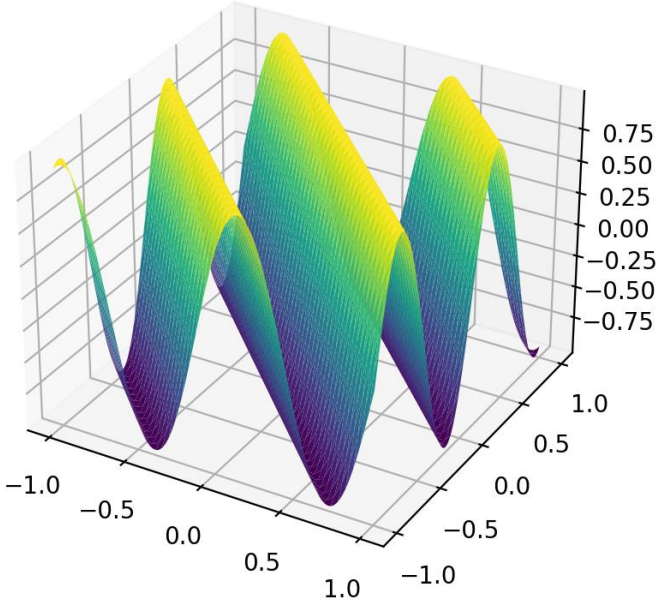


•

Sinusoid (corrugation)

$$s(x,y) = \sin(x_\alpha f)$$

$$\mathrm{x}_\alpha(x,y) = \mathrm{x} \cdot \cos(\alpha) - y \cdot \sin(\alpha)$$

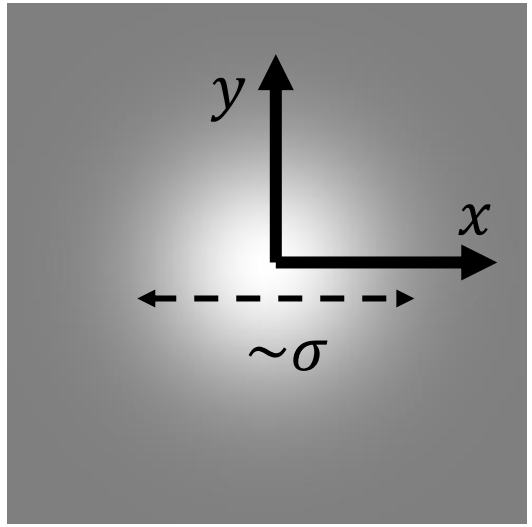


Gabor filter

Sensitive to edges of selected scale and orientation

Gaussian

$$g(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$$

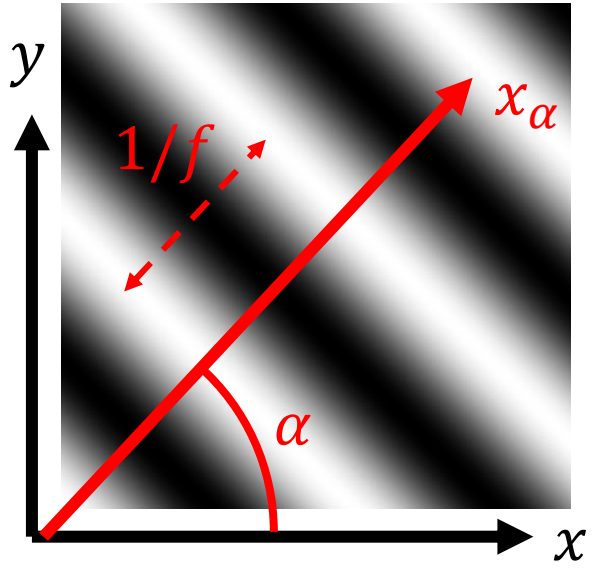


$\bullet$

Sinusoid (corrugation)

$$s(x,y) = \sin(x_\alpha f)$$

$$\mathrm{x}_\alpha(x,y) = \mathrm{x} \cdot \cos(\alpha) - y \cdot \sin(\alpha)$$

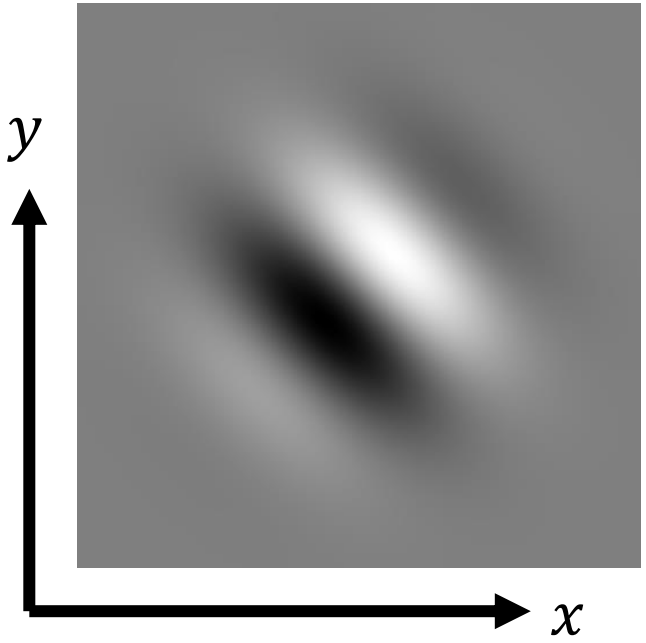


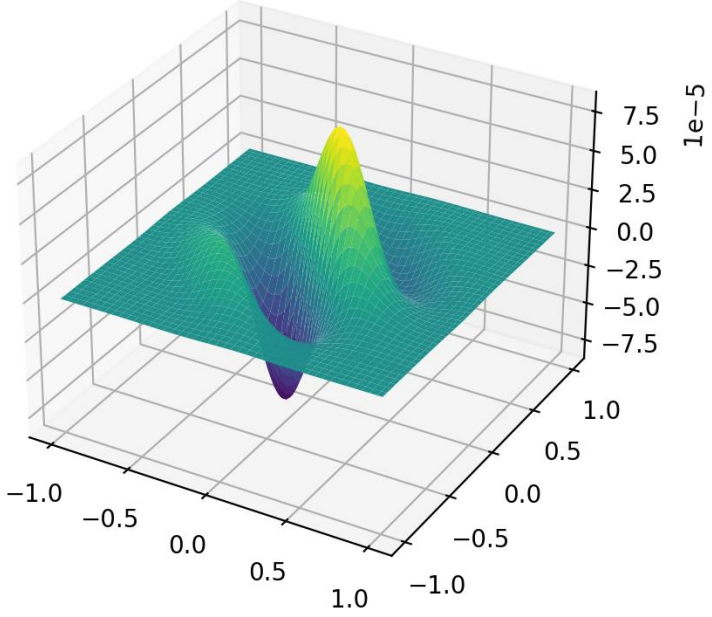
Gabor filter

Sensitive to edges of selected scale and orientation

Gabor filter

$$f(x, y) = g(x, y) \cdot s(x, y)$$

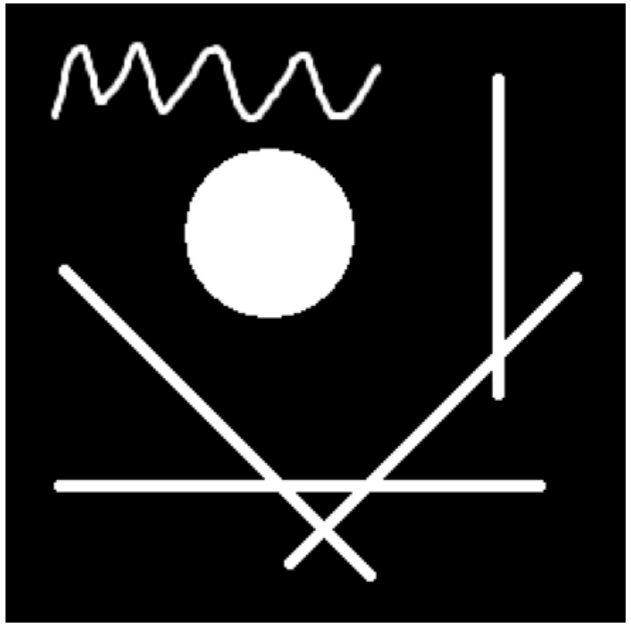




Example

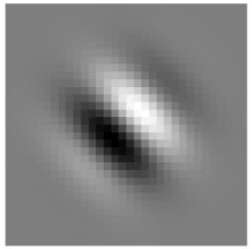
Detect diagonal edges using Gabor filter.

Correlation



Input image

$\star$



Gabor filter

$\alpha = 45° \quad f = 1/10 \text{ px}$

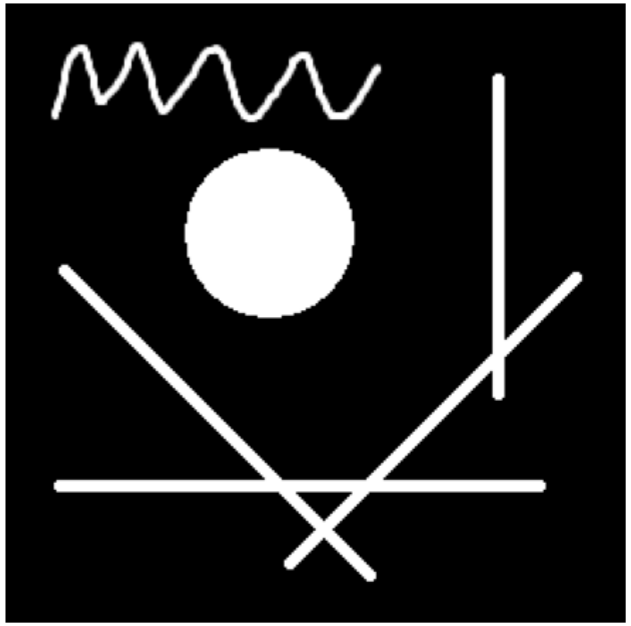
$=$

Filter response

Example

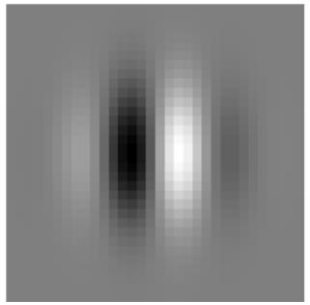
Detect vertical edges using Gabor filter.

Correlation



Input image

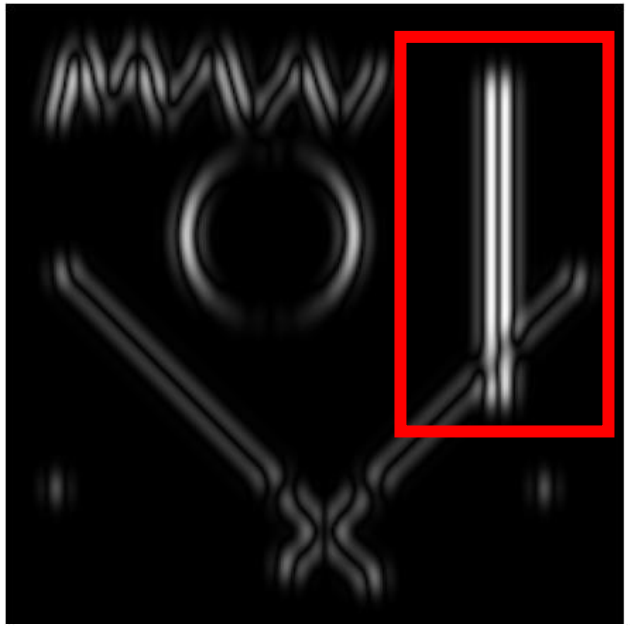
$\star$



Gabor filter

$\alpha = 0° \quad f = 1/10 \text{ px}$

$=$

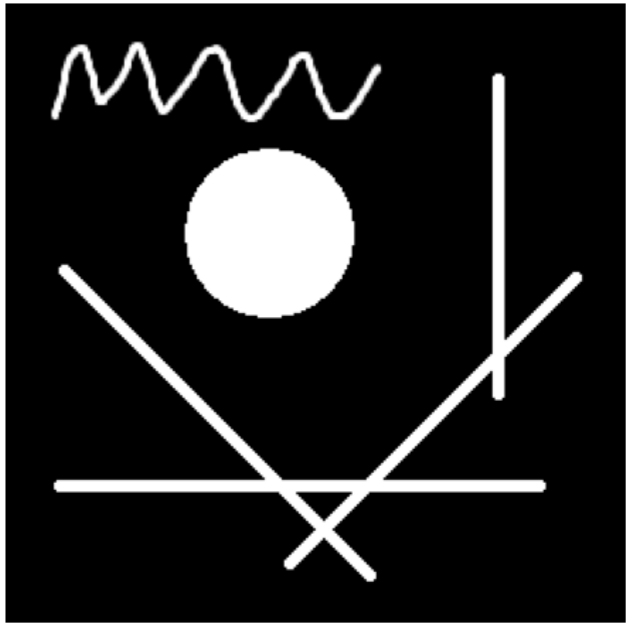


Filter response

Example

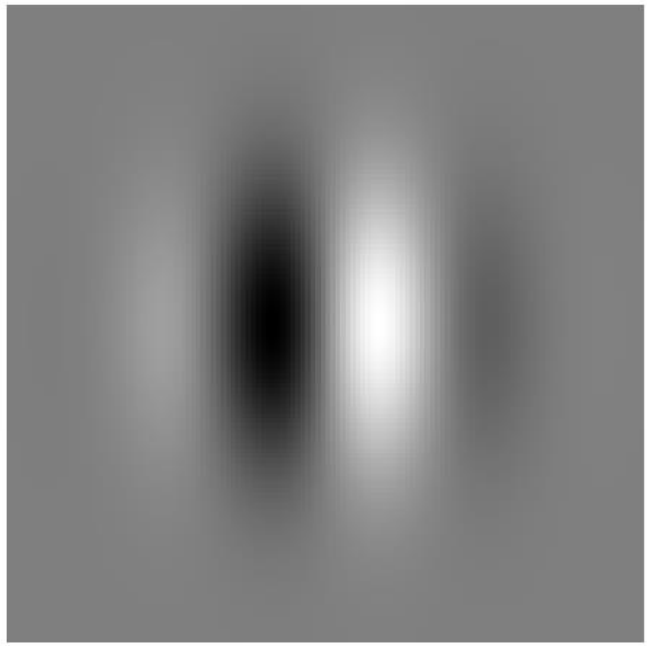
Detect large vertical “edges” using Gabor filter.

Correlation



Input image

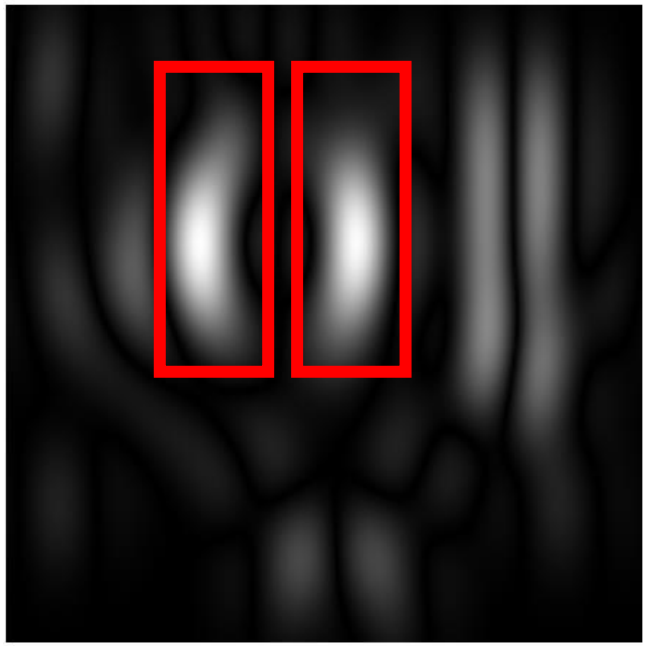
$\star$



Gabor filter

$\alpha = 0° \quad f = 1/80 \text{ px}$

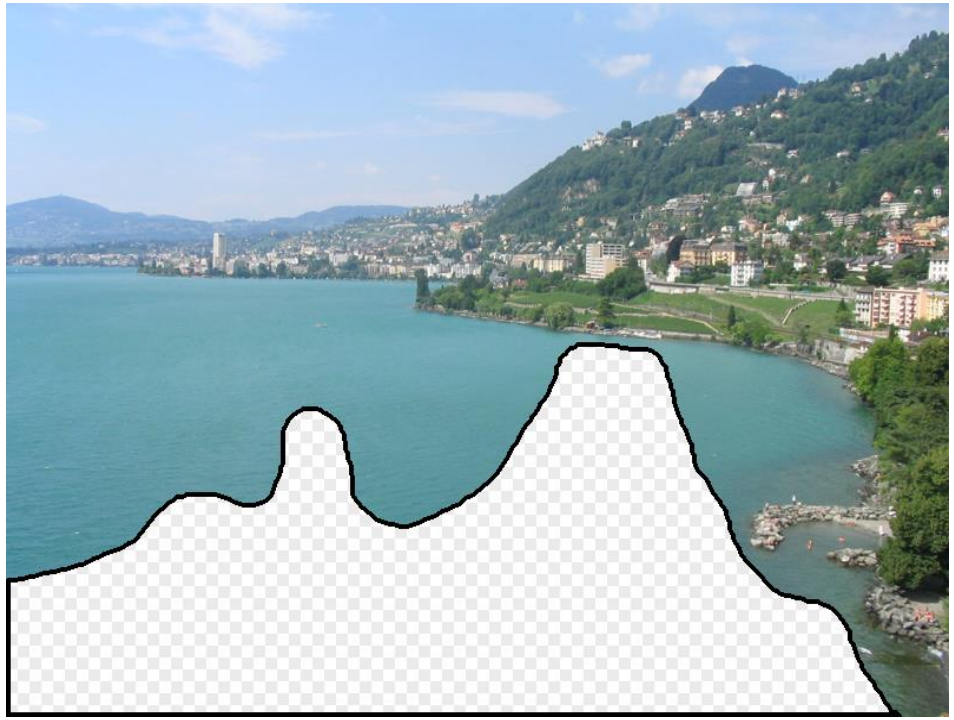
$=$



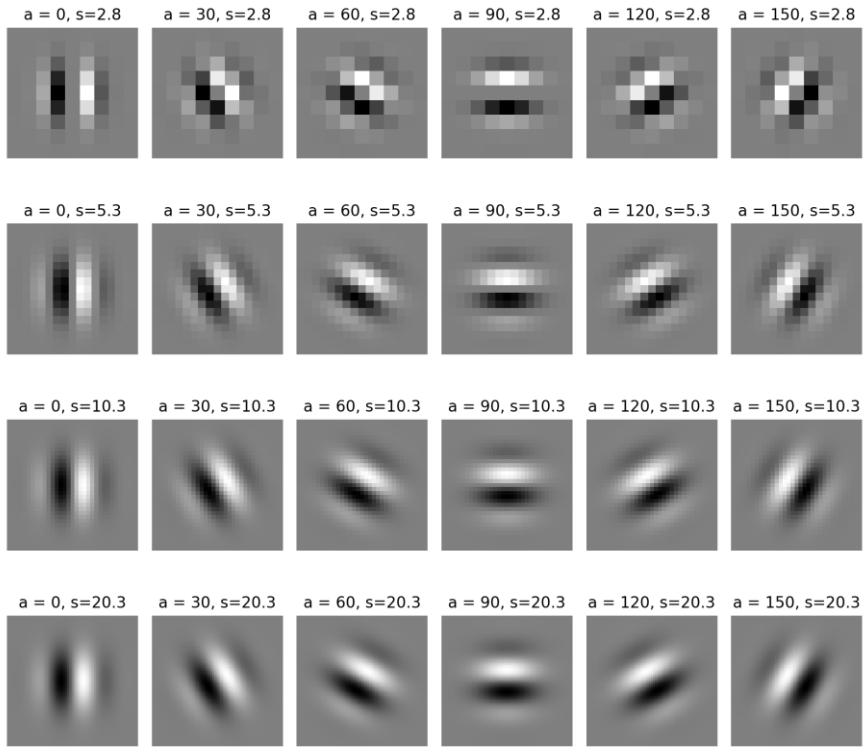
Filter response

Step 2: Image descriptor

I) Detect edges at 5 scales x 6 orientations.



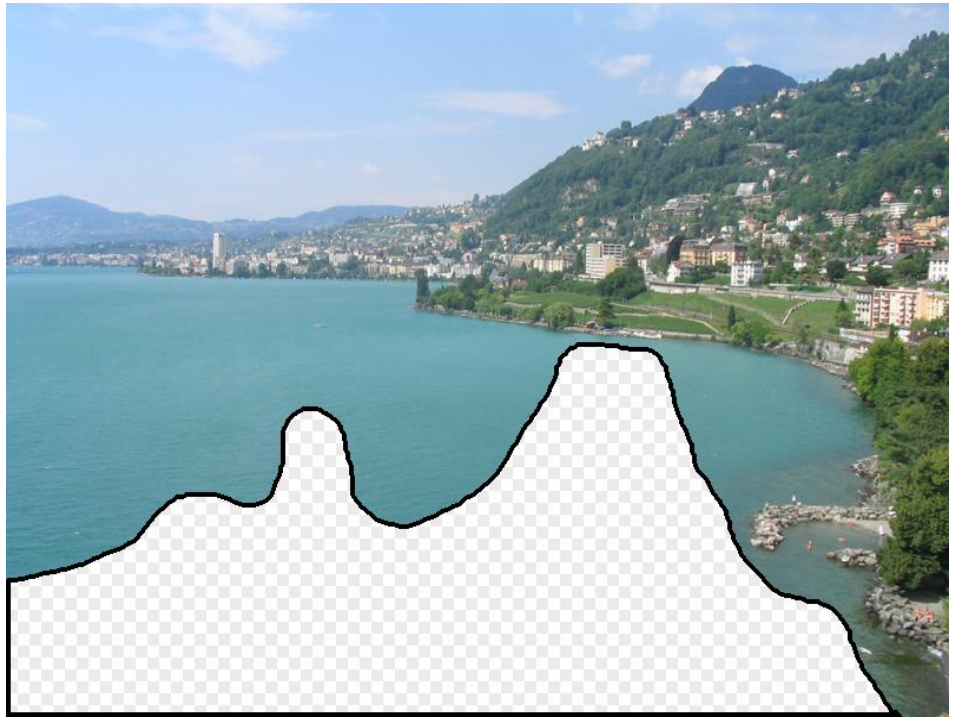
★

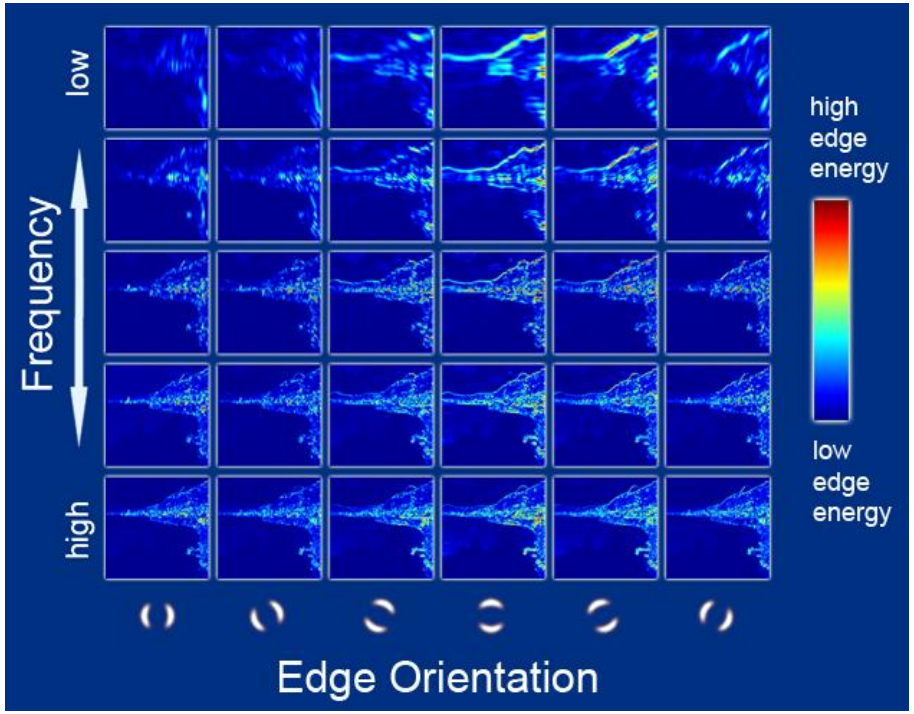


James Hays and Alexei A. Efros: Scene Completion Using Millions of Photographs. SIGGRAPH 2007.

Step 2: Image descriptor

I) Detect edges at 5 scales x 6 orientations.

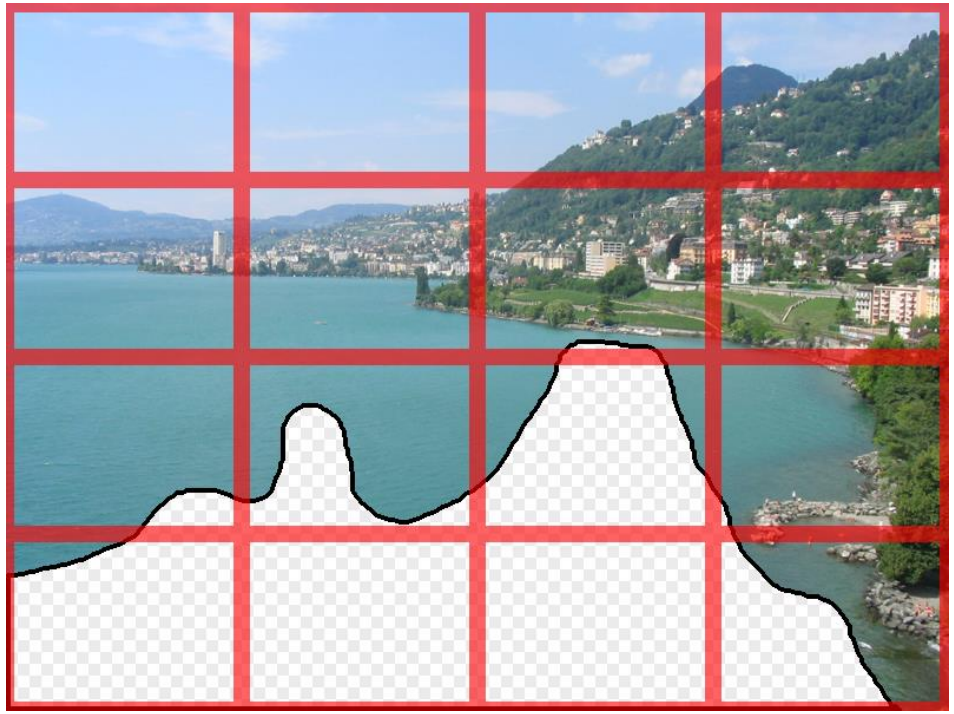


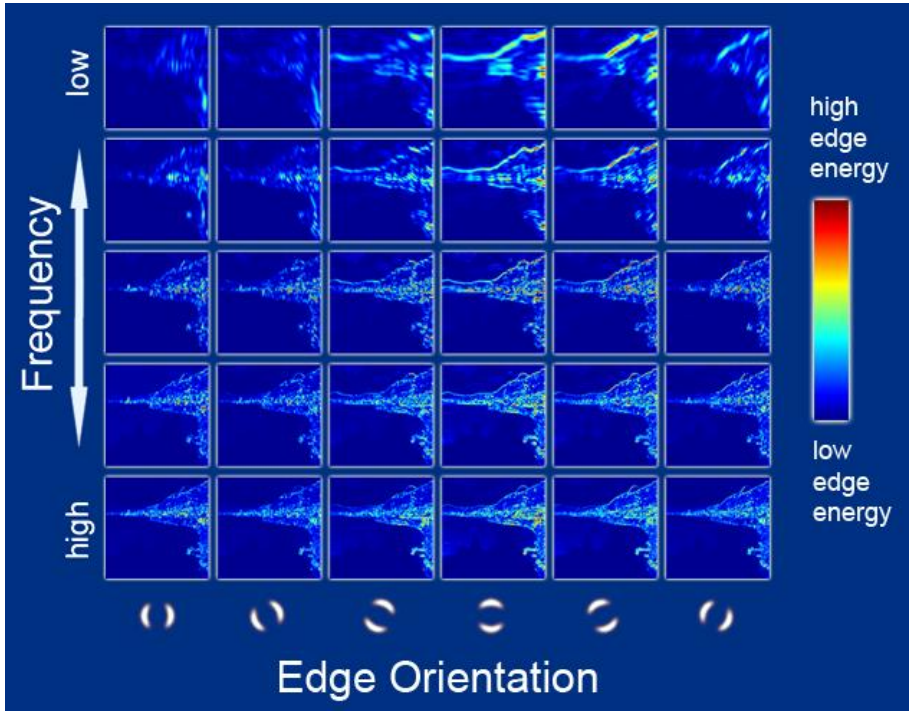


James Hays and Alexei A. Efros: Scene Completion Using Millions of Photographs. SIGGRAPH 2007.

Step 2: Image descriptor

II) Subdivide image to 4x4 blocks.

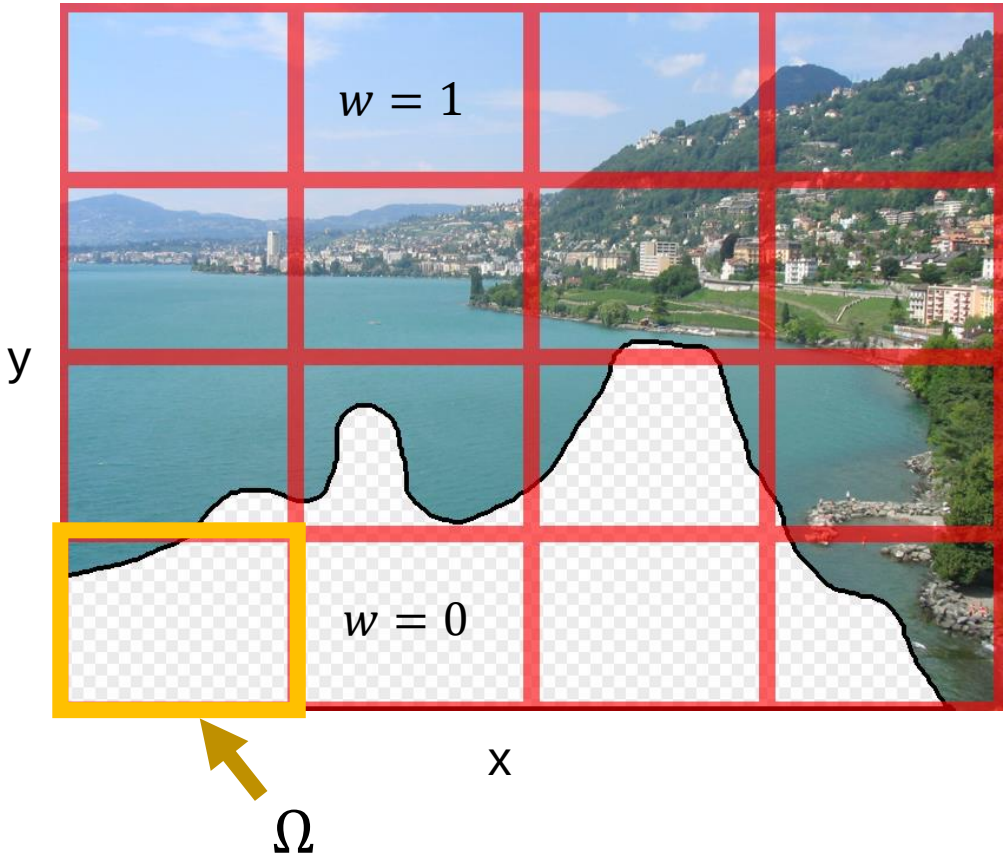


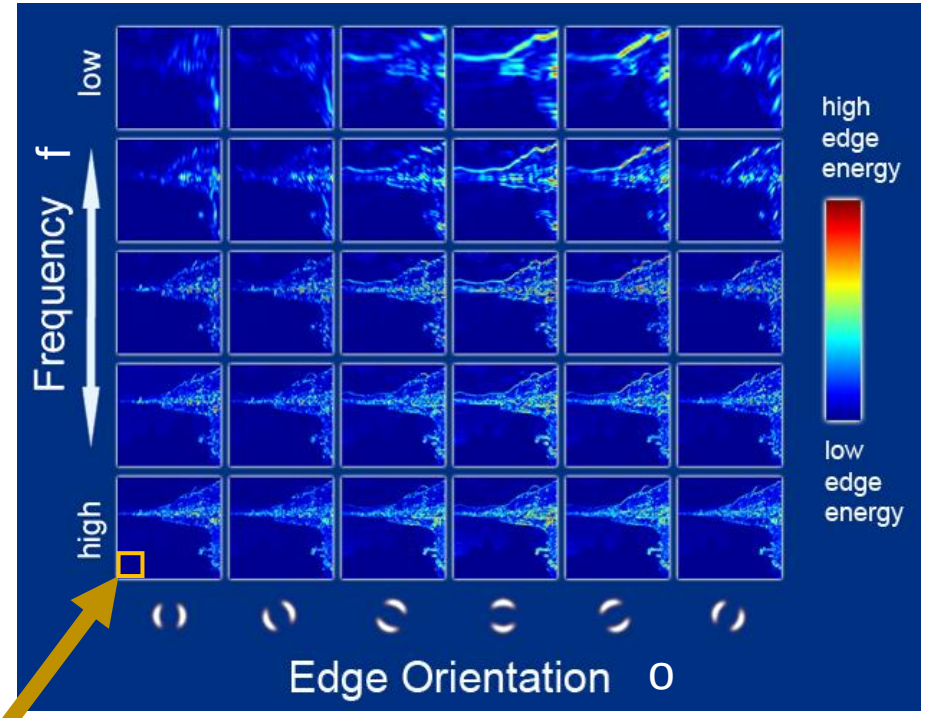


James Hays and Alexei A. Efros: Scene Completion Using Millions of Photographs. SIGGRAPH 2007.

Step 2: Image descriptor

III) Compute mean edge intensity in each block => histogram(s)





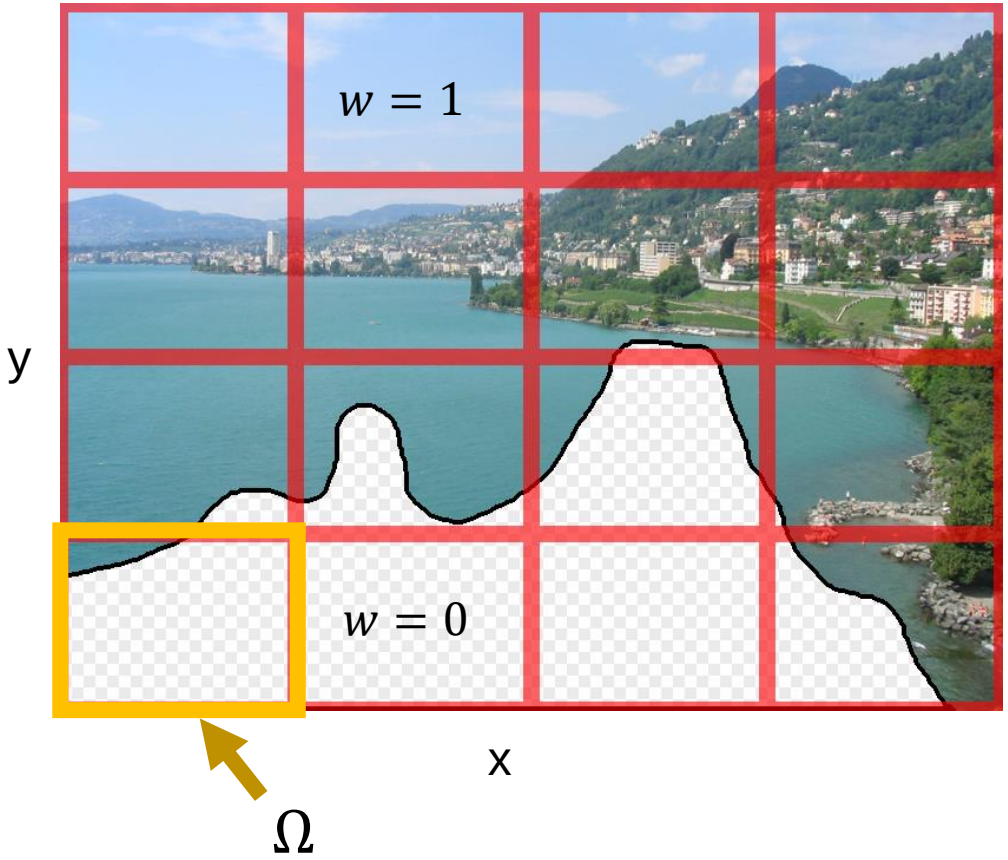
$$\{e_i\} \in \Omega_{x,y,o,f}$$

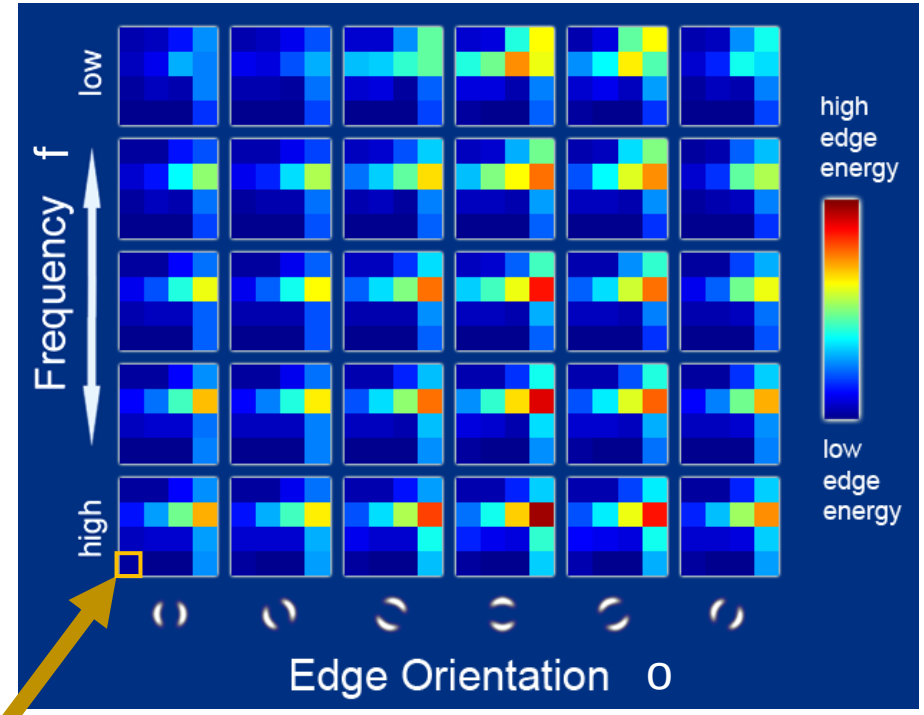
$$h_{x,y,o,f} = \frac{\sum_{i \in \Omega_{x,y,o,f}} w_i \cdot e_i}{\sum_{i \in \Omega_{x,y,o,f}} w_i}$$

James Hays and Alexei A. Efros: Scene Completion Using Millions of Photographs. SIGGRAPH 2007.

Step 2: Image descriptor

III) Compute mean edge intensity in each block => histogram(s)





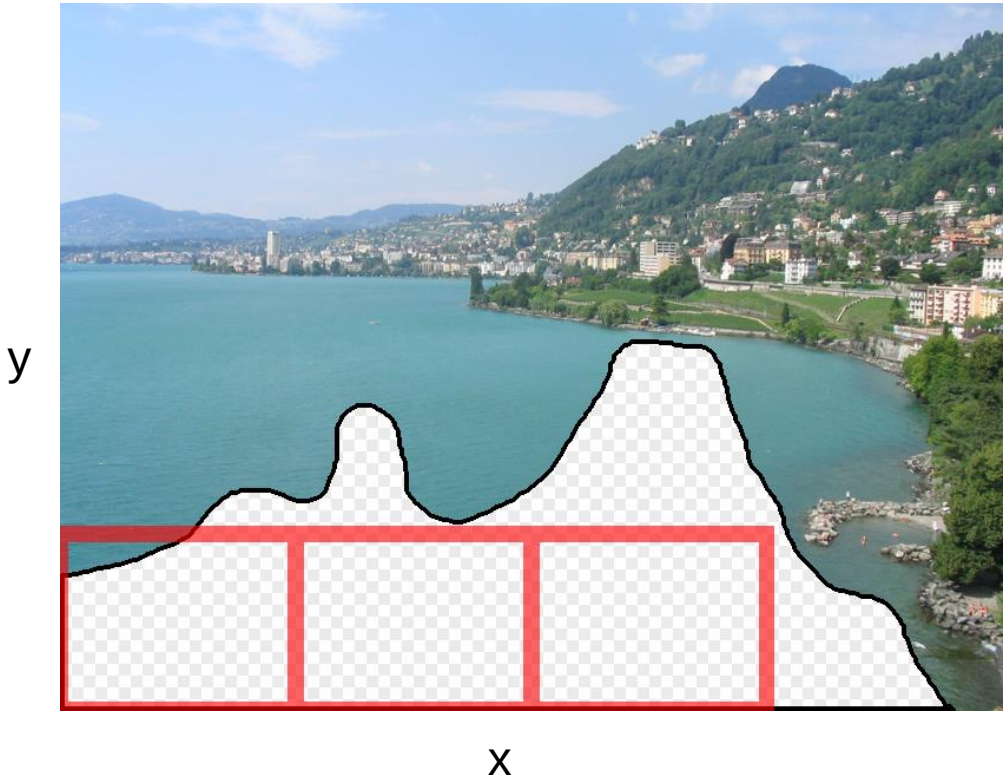
$h_{x,y,o,f}$

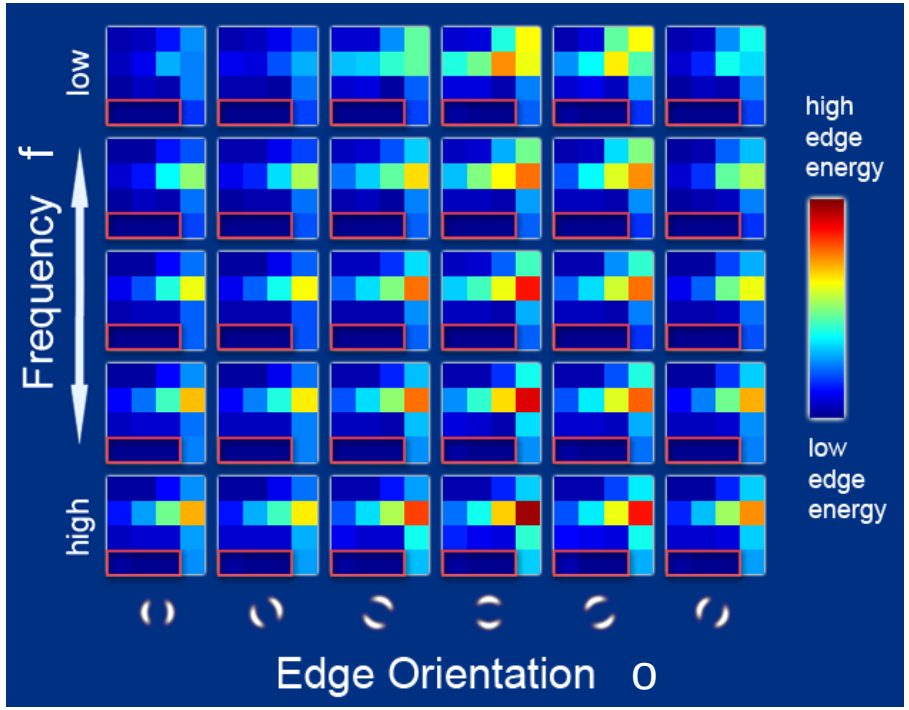
$$h_{x,y,o,f} = \frac{\sum_{i \in \Omega_{x,y,o,f}} w_i \cdot e_i}{\sum_{i \in \Omega_{x,y,o,f}} w_i}$$

James Hays and Alexei A. Efros: Scene Completion Using Millions of Photographs. SIGGRAPH 2007.

Step 2: Image descriptor

Note: The blocks corresponding to the holes are empty.

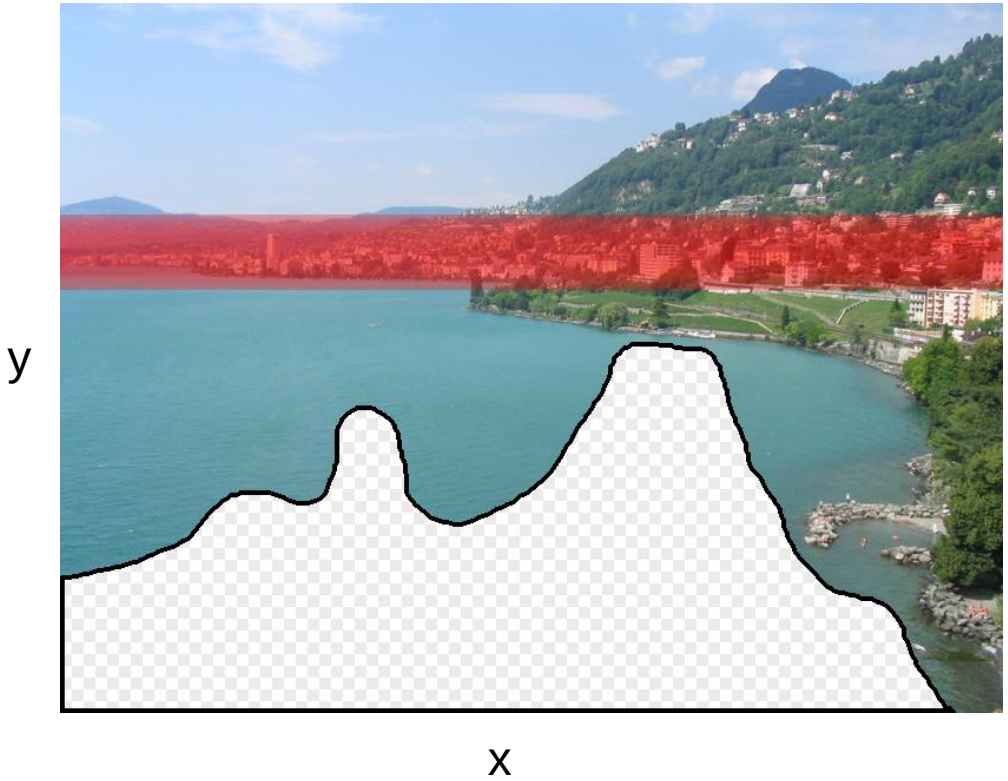


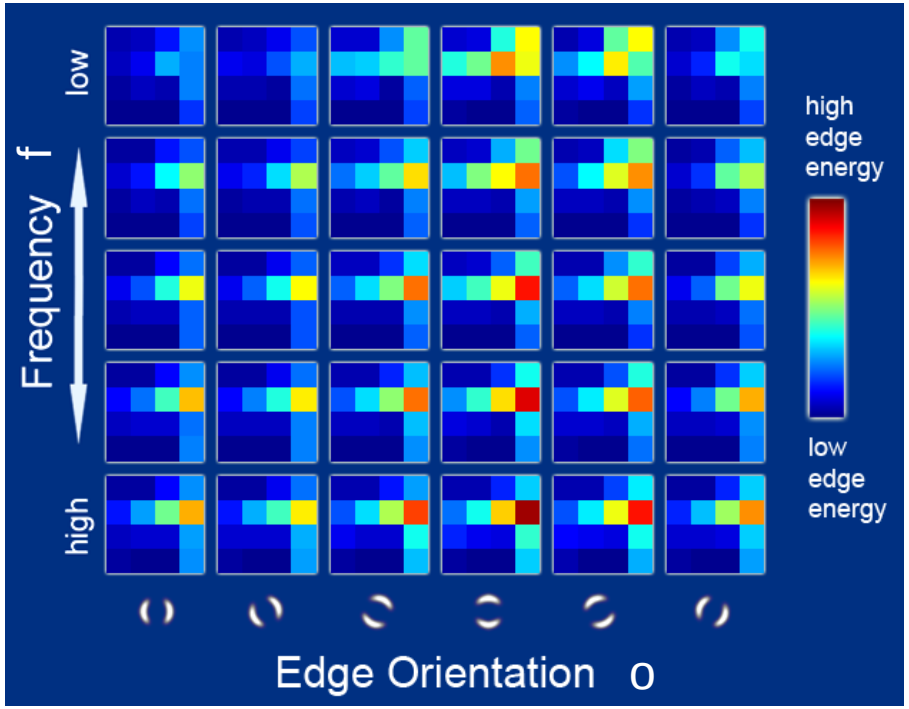


James Hays and Alexei A. Efros: Scene Completion Using Millions of Photographs. SIGGRAPH 2007.

Step 2: Image descriptor

Note: The blocks with most complex patterns have the strongest signal.

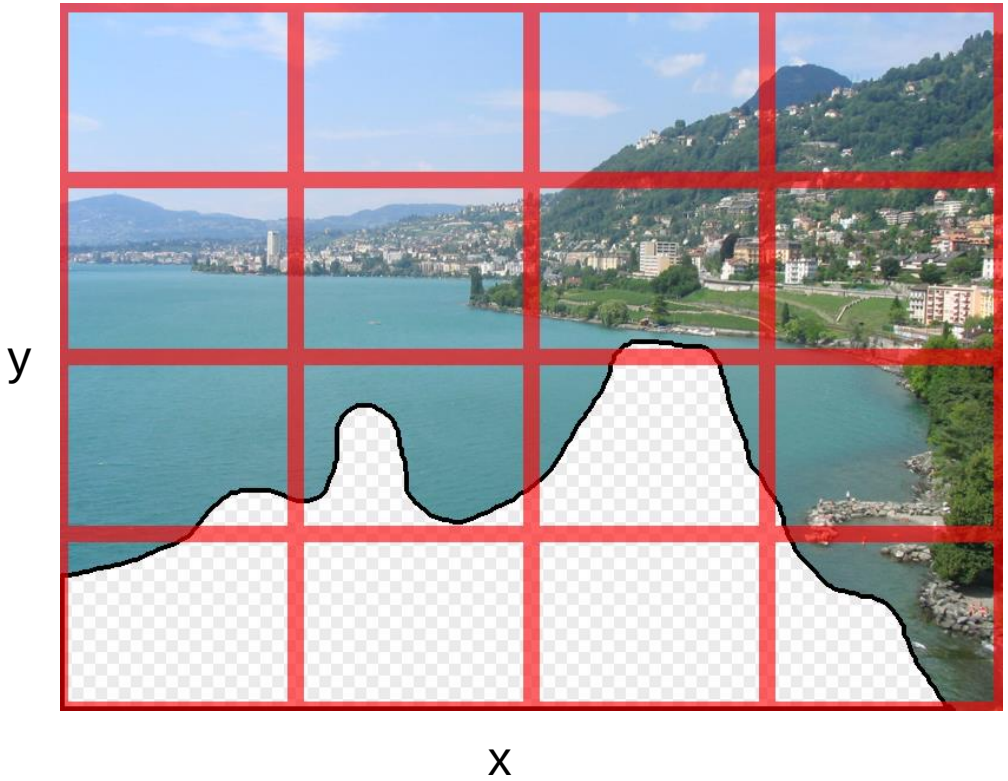


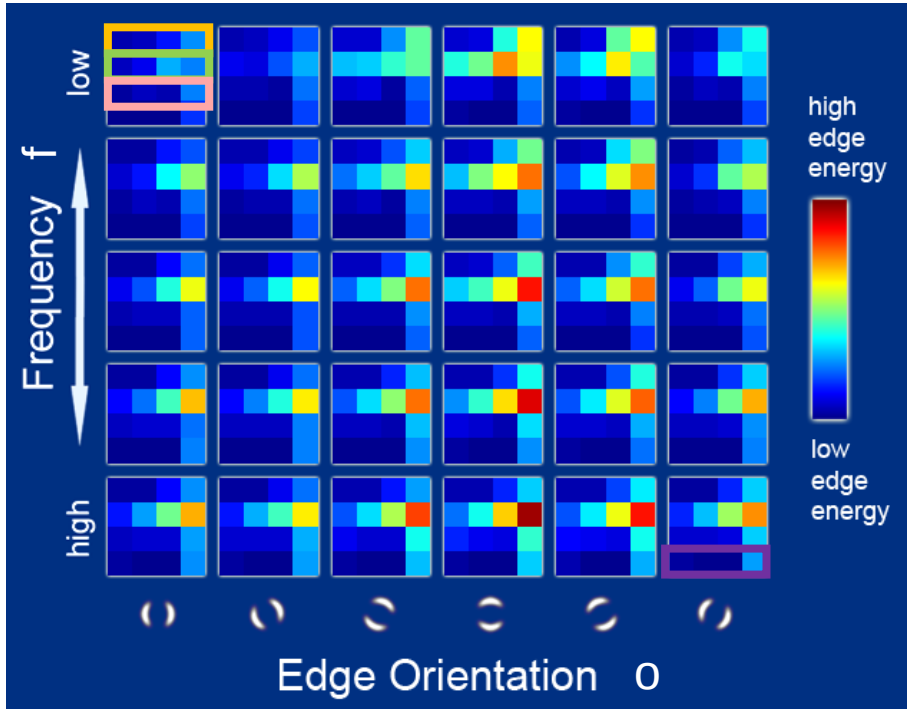


James Hays and Alexei A. Efros: Scene Completion Using Millions of Photographs. SIGGRAPH 2007.

Step 2: Image descriptor

IV) Concatenate all sub-histograms to a long vector.





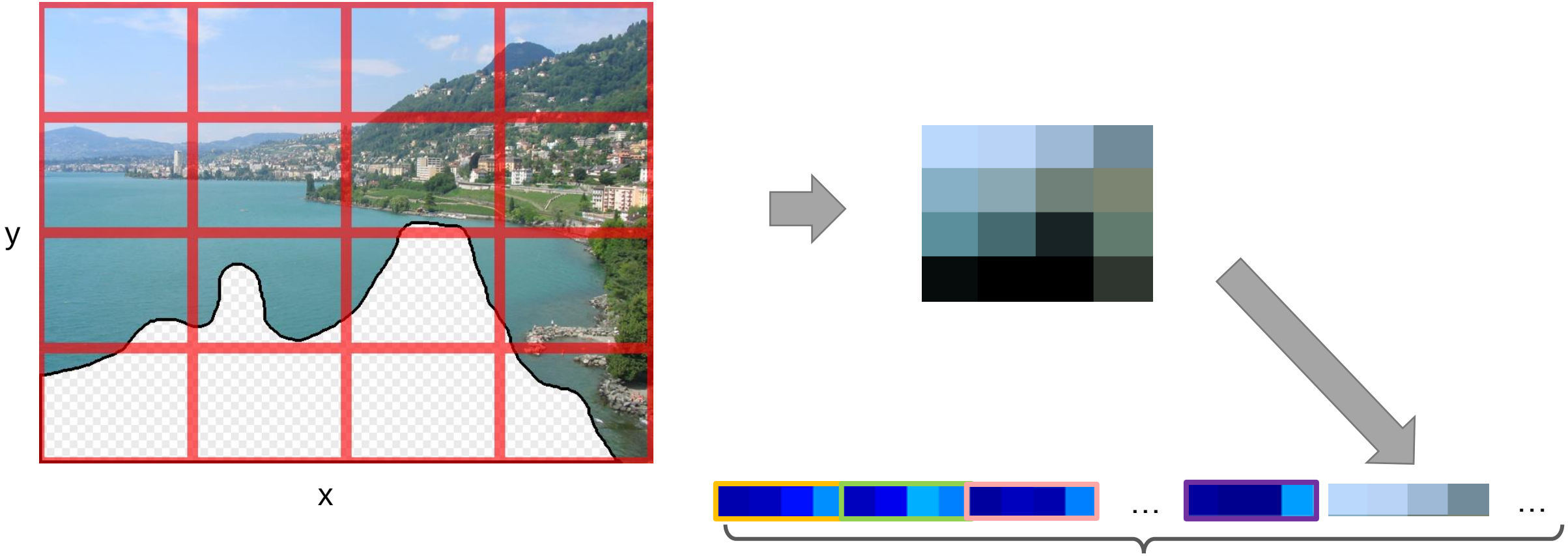
4 x 4 x 6 x 5 = 480 long GIST descriptor

James Hays and Alexei A. Efros: Scene Completion Using Millions of Photographs. SIGGRAPH 2007.

Torralba, Antonio, et al. "Context-based vision system for place and object recognition." Computer Vision, IEEE International Conference on. Vol. 2. IEEE Computer Society, 2003.

Step 2: Image descriptor

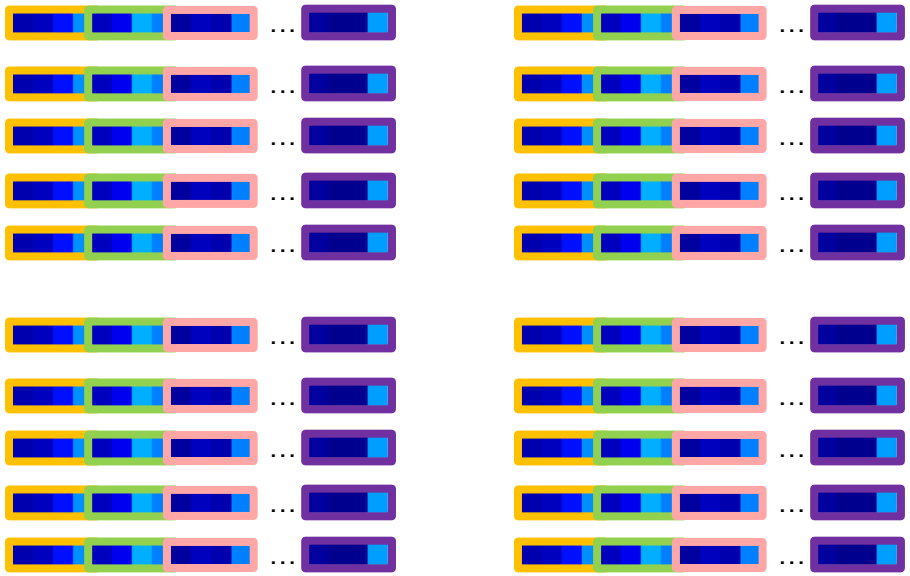
V) Concatenate a downsampled version of the image.



James Hays and Alexei A. Efros: Scene Completion Using Millions of Photographs. SIGGRAPH 2007.

Step 3: Image Collection

Compute GIST descriptor for every image of a collection D.



James Hays and Alexei A. Efros: Scene Completion Using Millions of Photographs. SIGGRAPH 2007.

Step 3: Image Collection

We need **millions of images** to guarantee that there is at least one image very similar to ours.

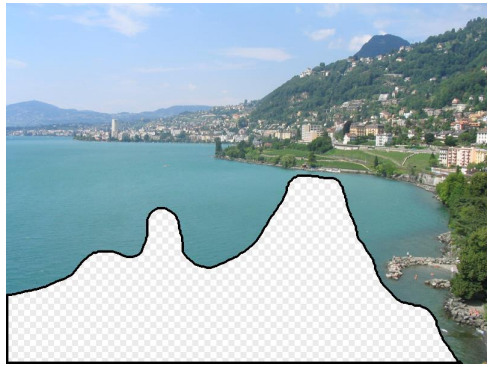
James Hays and Alexei A. Efros: Scene Completion Using Millions of Photographs. SIGGRAPH 2007.

Step 4: Find k-nearest matches

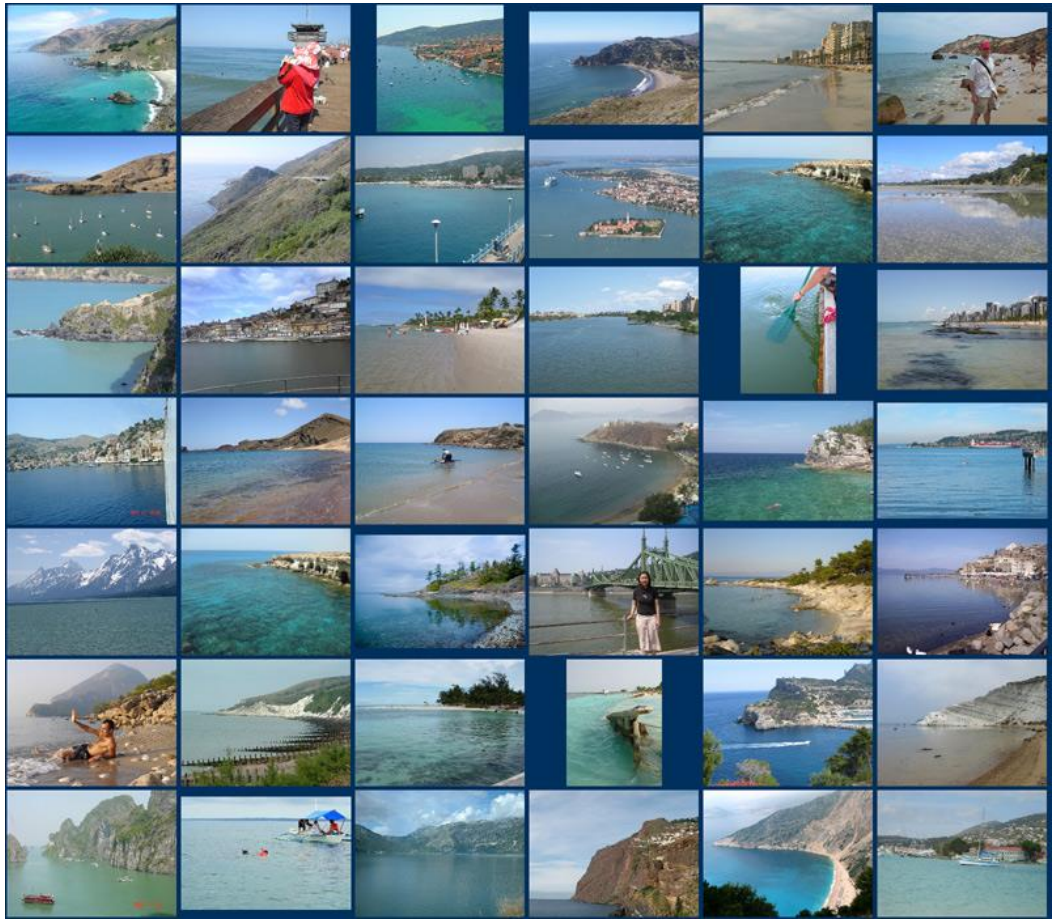
200 images with the smallest SSD distance between the descriptors.

$$SSD(\boldsymbol{a}, \boldsymbol{b}) = \sum_i (a_i - b_i)^2$$

Sum of Squared Differences



Input



k-nearest matches

James Hays and Alexei A. Efros: Scene Completion Using Millions of Photographs. SIGGRAPH 2007.

Step 5: Select and align

Select a collection <u>image</u> I_C and <u>transformation</u> $\boldsymbol{t}$ which best matches input image boundary I_B.

$$\{I_C, \boldsymbol{t}\} = \underset{I \in \mathrm{D},\, \boldsymbol{t}}{argmin}\, SSD(I, \mathbf{t}(I_B))$$

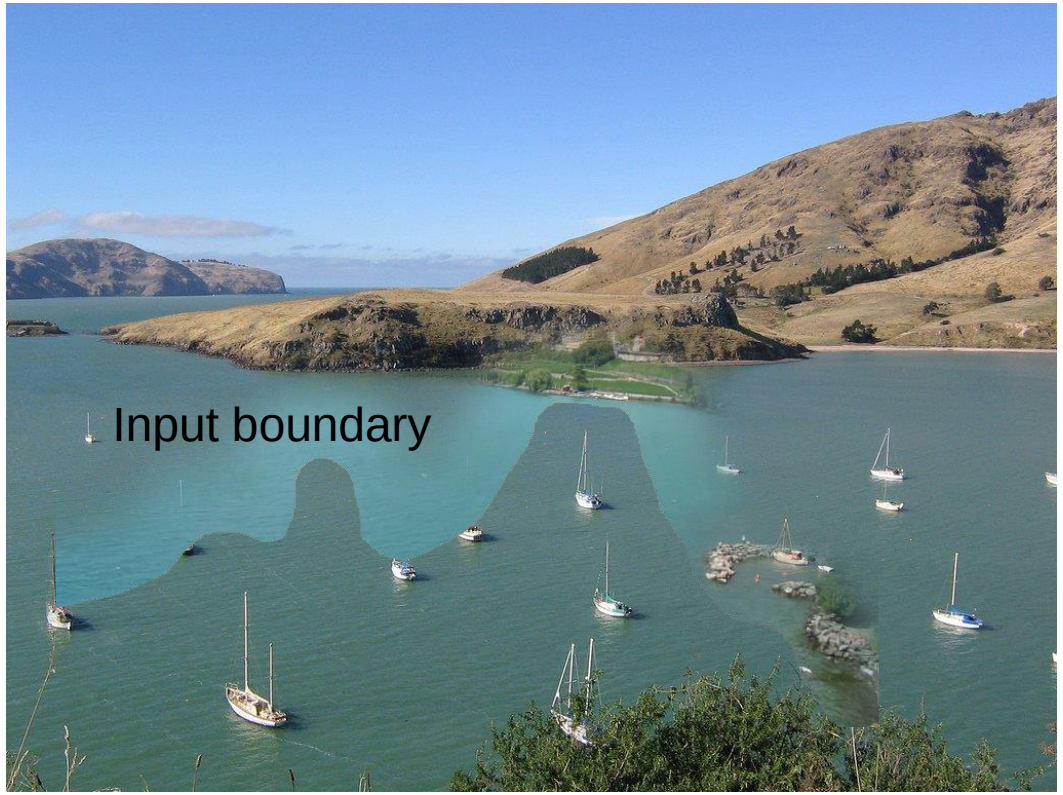
Input boundary

Selected image from the collection

James Hays and Alexei A. Efros: Scene Completion Using Millions of Photographs. SIGGRAPH 2007.

Step 5: Select and align

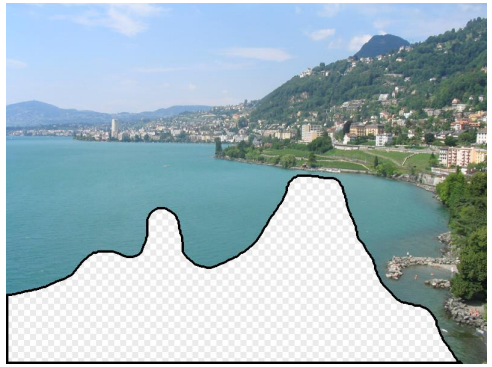
Select a collection <u>image</u> I_C and <u>transformation</u> $\boldsymbol{t}$ which best matches input image boundary I_B.



Selected image from the collection

James Hays and Alexei A. Efros: Scene Completion Using Millions of Photographs. SIGGRAPH 2007.

Step 6: Merge using Poisson blending

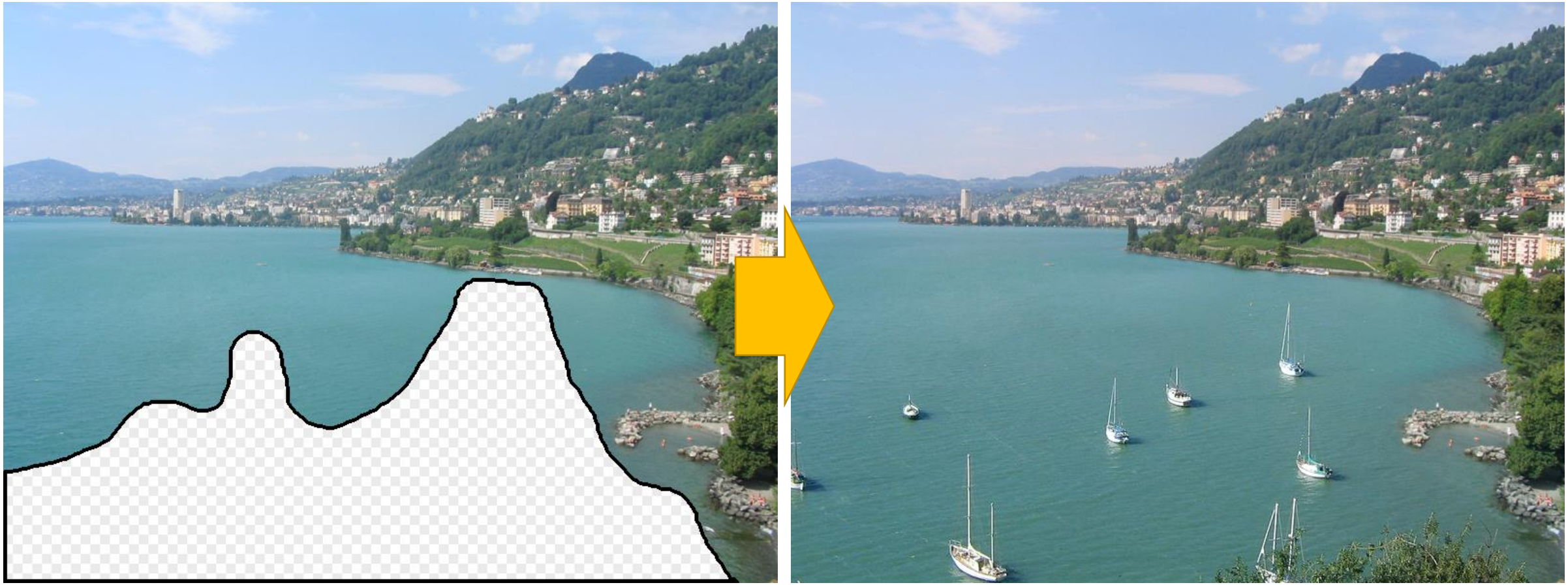


Input

Selected image from the collection

James Hays and Alexei A. Efros: Scene Completion Using Millions of Photographs. SIGGRAPH 2007.

Step 6: Merge the two images

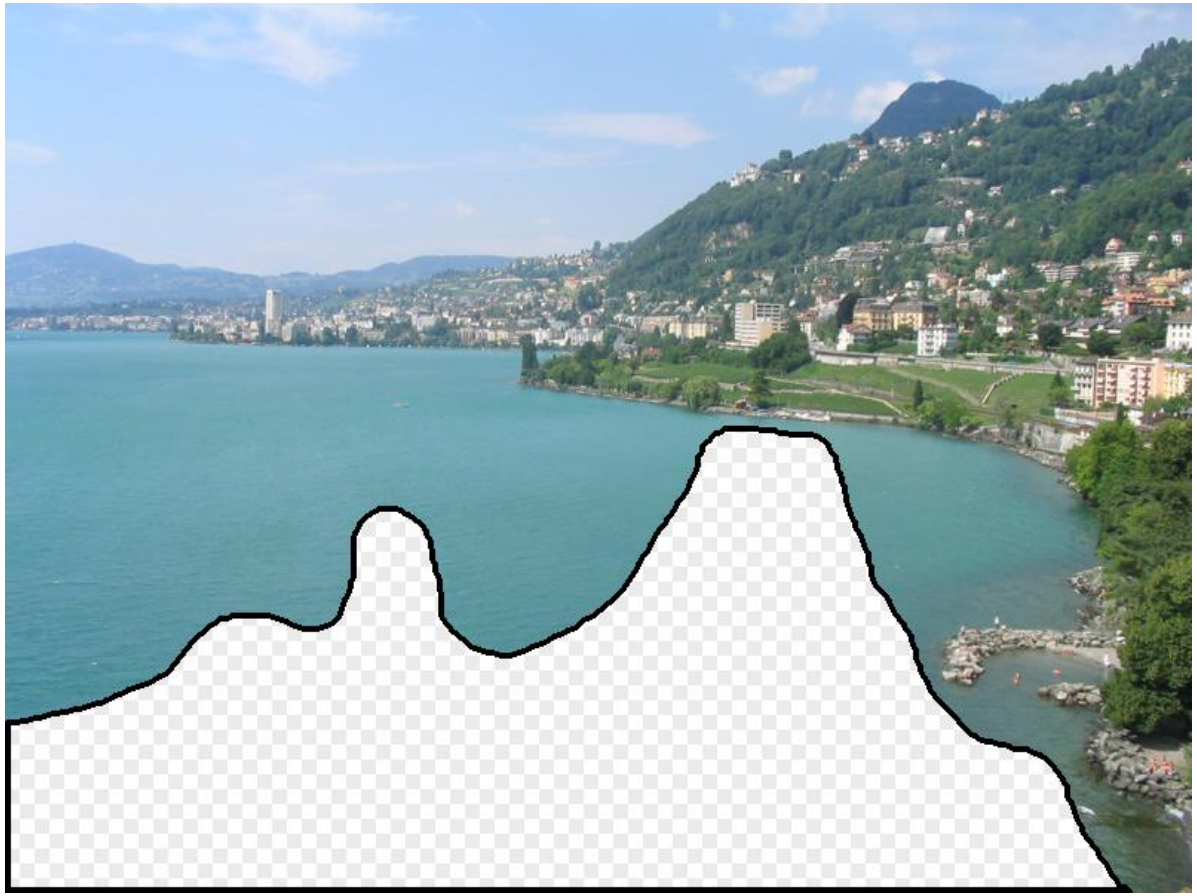


James Hays and Alexei A. Efros: Scene Completion Using Millions of Photographs. SIGGRAPH 2007.

Alternative image choices

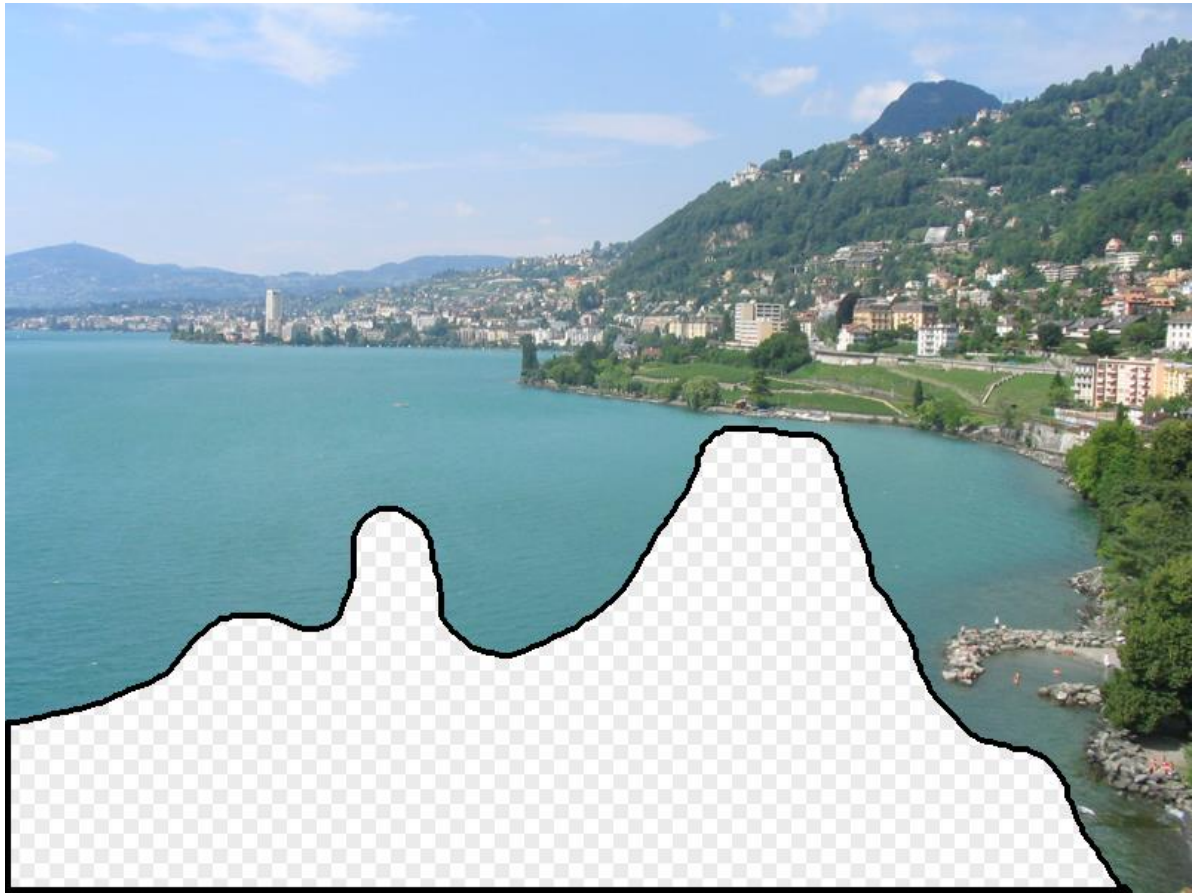
James Hays and Alexei A. Efros: Scene Completion Using Millions of Photographs. SIGGRAPH 2007.

Alternative image choices



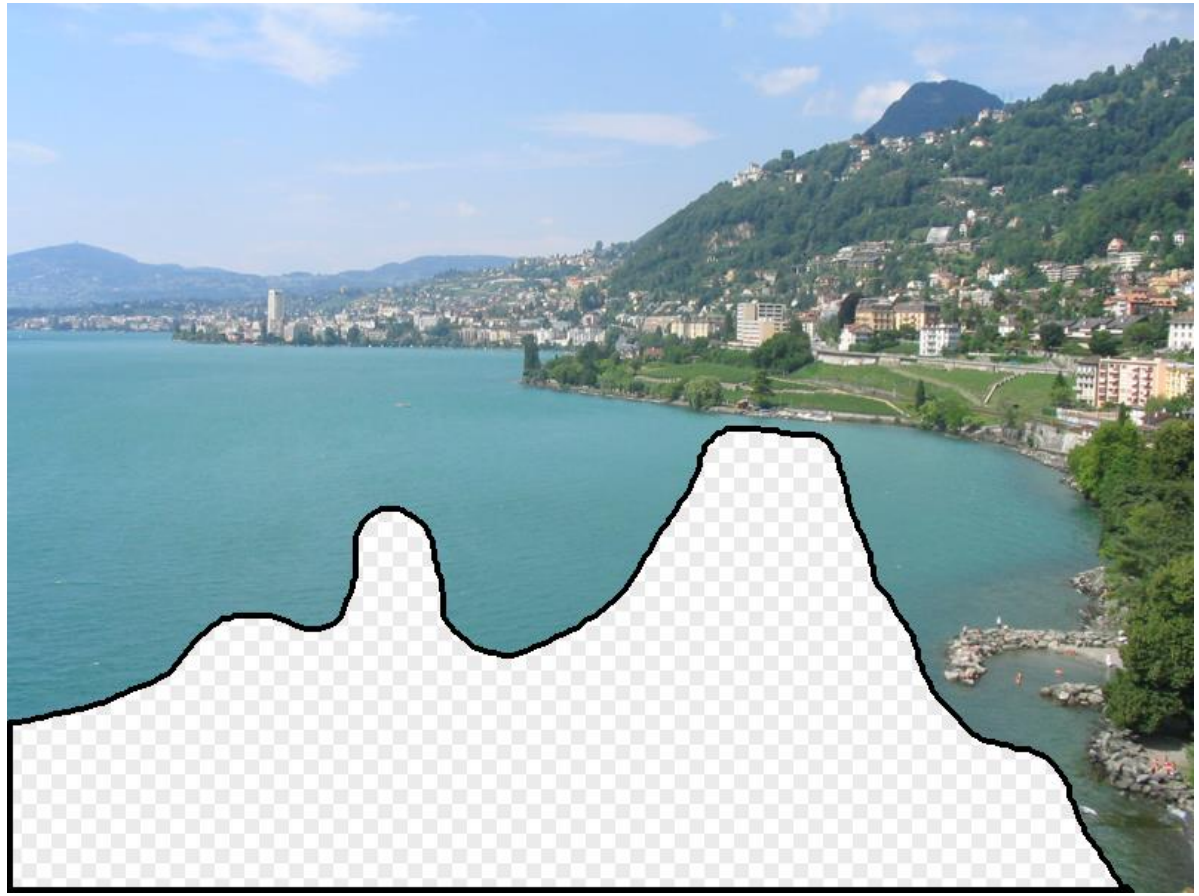
James Hays and Alexei A. Efros: Scene Completion Using Millions of Photographs. SIGGRAPH 2007.

Alternative image choices



James Hays and Alexei A. Efros: Scene Completion Using Millions of Photographs. SIGGRAPH 2007.

Alternative image choices



James Hays and Alexei A. Efros: Scene Completion Using Millions of Photographs. SIGGRAPH 2007.

Collection sampling

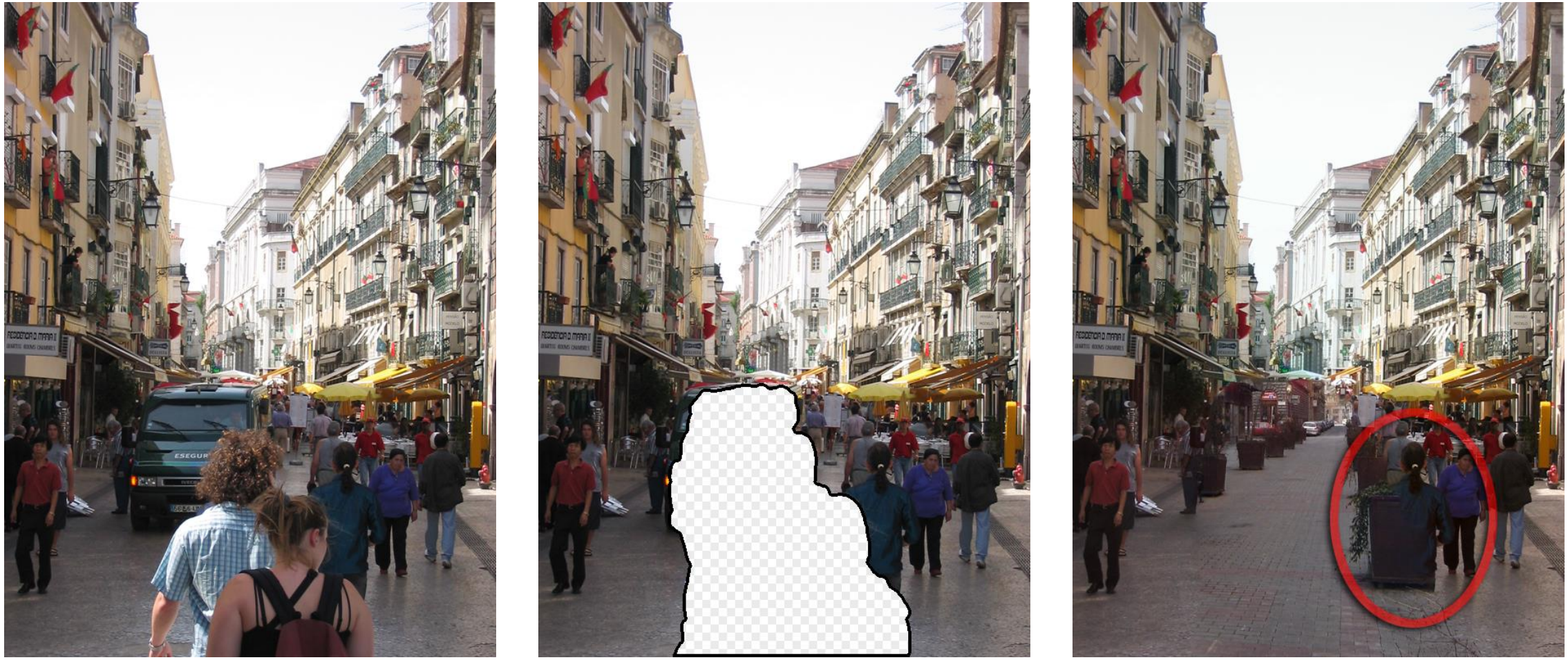
Good

- Can fill large holes.
- Introduces new content.

Bad

- Data intense.
- Compute intense.
- Limited to what is in the collection.

Failure case



James Hays and Alexei A. Efros: Scene Completion Using Millions of Photographs. SIGGRAPH 2007.

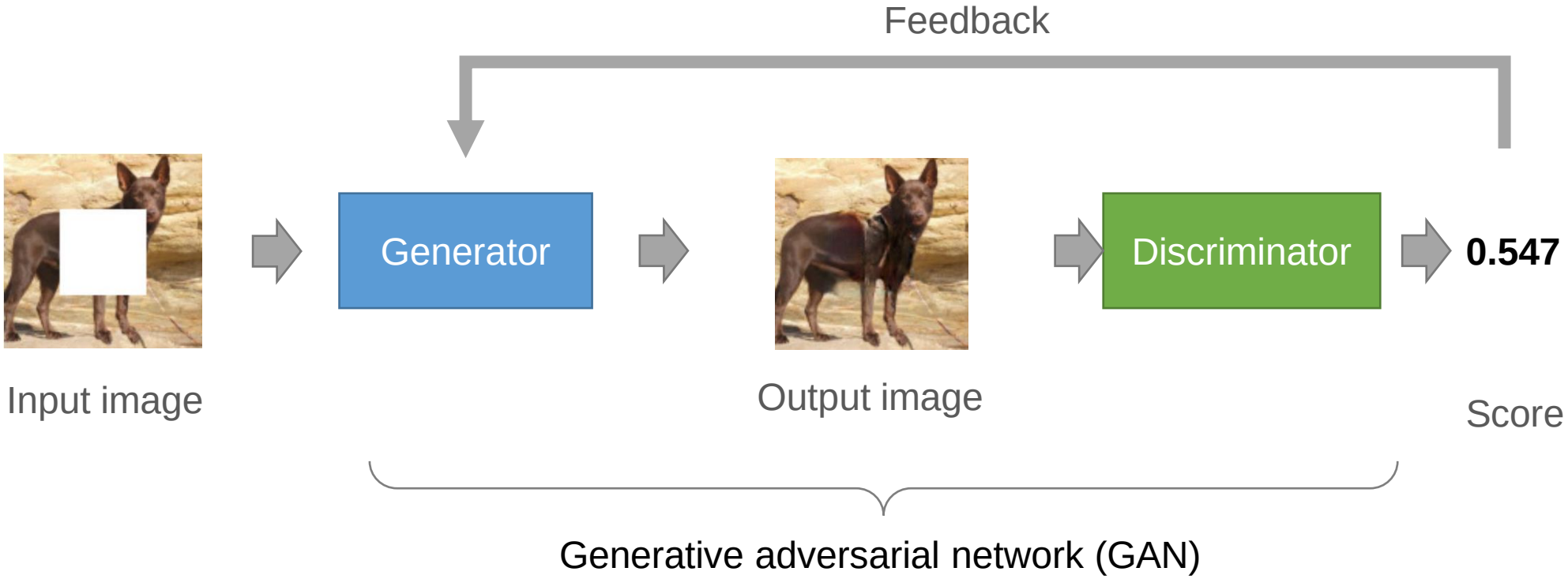
Algorithms

	Statistic		Prior
1. Pull-Push	Simple: Mean color		Single image
2. PDE inpainting	Simple: Gradients		Single image
3. Exemplar inpainting	Medium: Patterns		Single image
4. Collection sampling	Medium: Patterns		Collection
5. Neural inpainting	Complex: Multi-scale		Collection

Neural inpainting using Generative Adversarial Networks (GANs)

Idea: Let a neural network do both jobs for us:

1. Fill the missing part => Generator.
2. Judge the result => Discriminator.

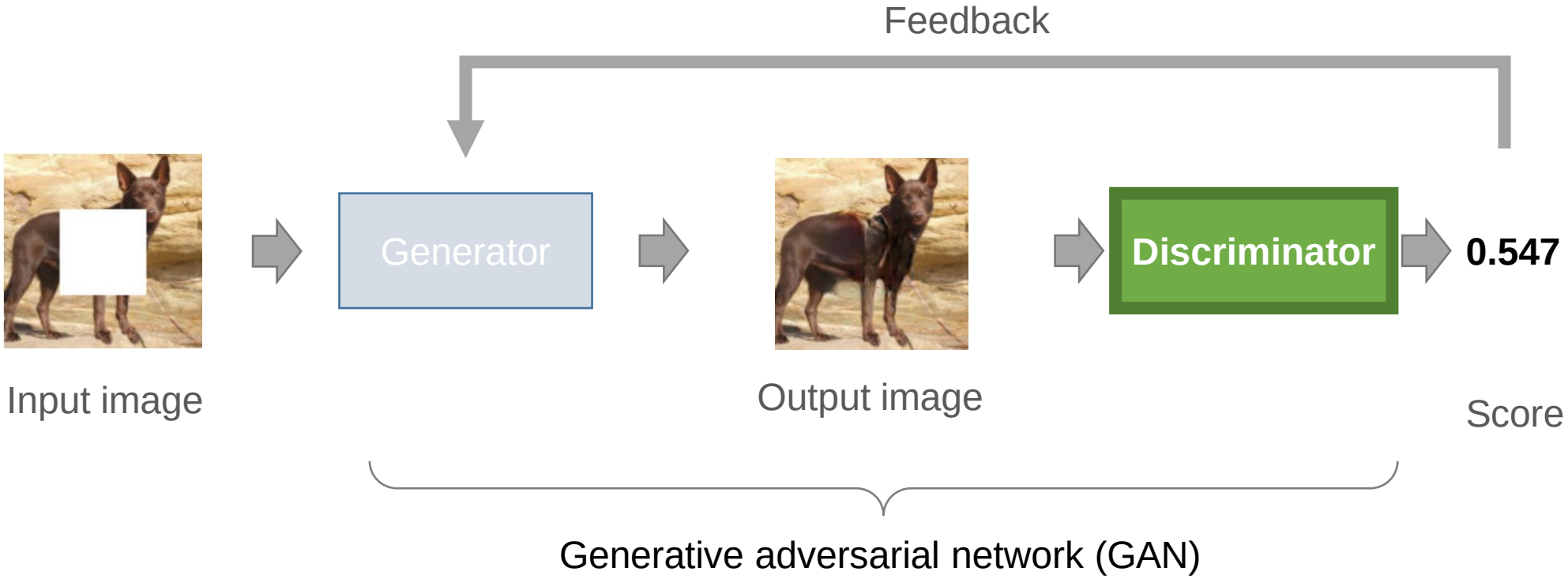


Iizuka, Satoshi, Edgar Simo-Serra, and Hiroshi Ishikawa. "Globally and locally consistent image completion." . SIGGRAPH 2017.

Neural inpainting using Generative Adversarial Networks (GANs)

Idea: Let a neural network do both jobs for us:

1. Fill the missing part => Generator.
2. Judge the result => Discriminator.



Iizuka, Satoshi, Edgar Simo-Serra, and Hiroshi Ishikawa. "Globally and locally consistent image completion." . SIGGRAPH 2017.

Discriminator

Uses complex **learned image statistics** to separate real and generated ("fake") images.

Credit: Osvaldo Bojanovich @ flickr

Real image from a collection

Discriminator → **0.894** *"Looks real"*

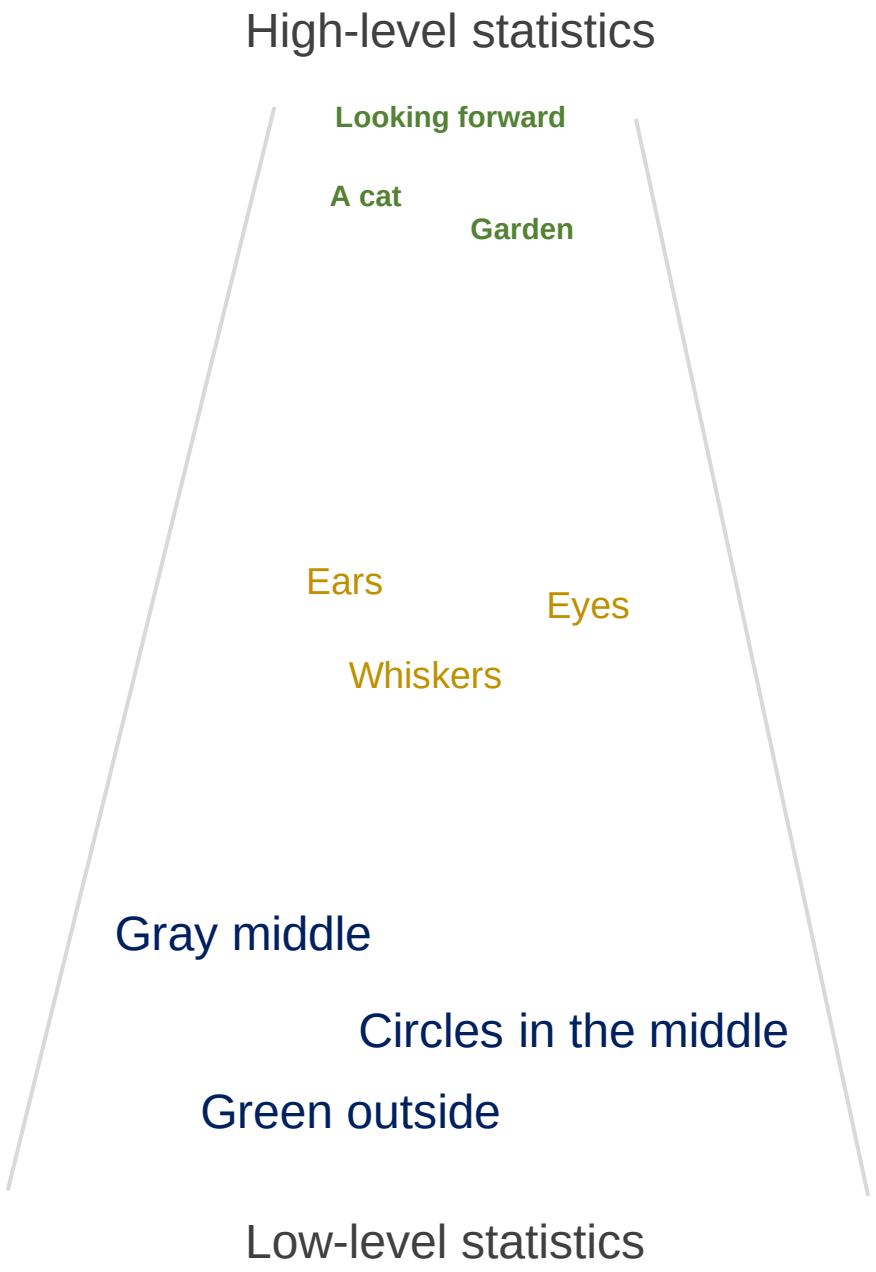
Inpainted image from the generator

Discriminator → **0.254** *"Looks fake"*

Discriminator

- Pyramid of image statistics extractors
  - Complex knowledge built from simple filters.
- Reasons about image consistency
  - Locally consistent
    "Does the fur look ok?"
  - Globally consistent
    "Are the ears on top of the head?"

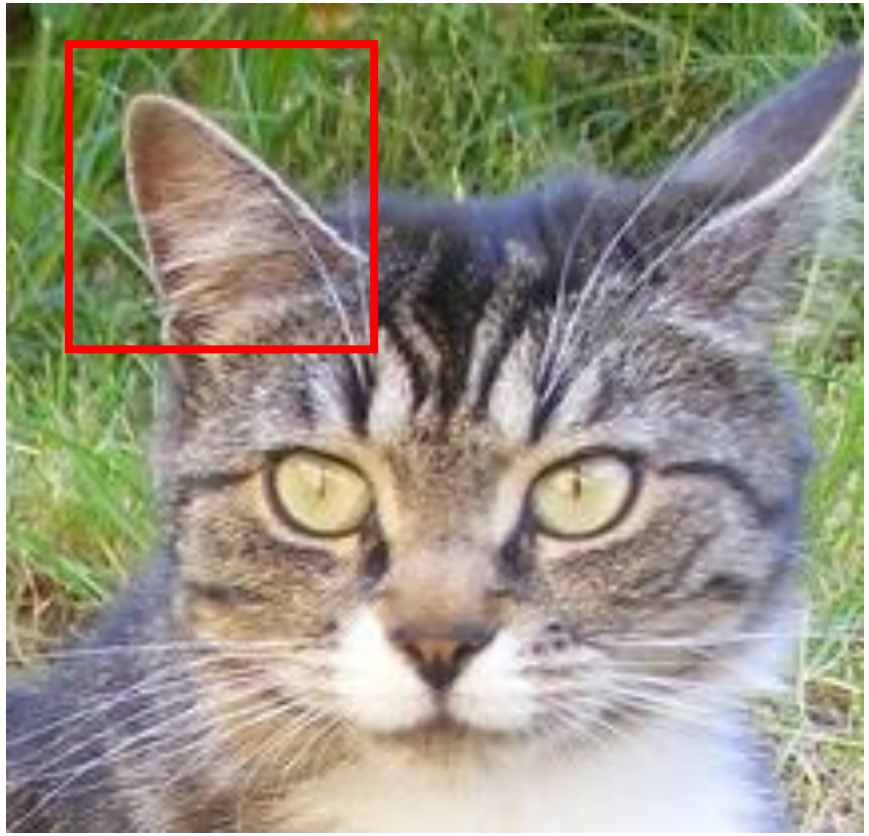
Credit: Andrew Wilkinson @ flickr



Example: Ear detector

Let us try to find ear-tip in the image manually.

Image



Credit: Andrew Wilkinson @ flickr

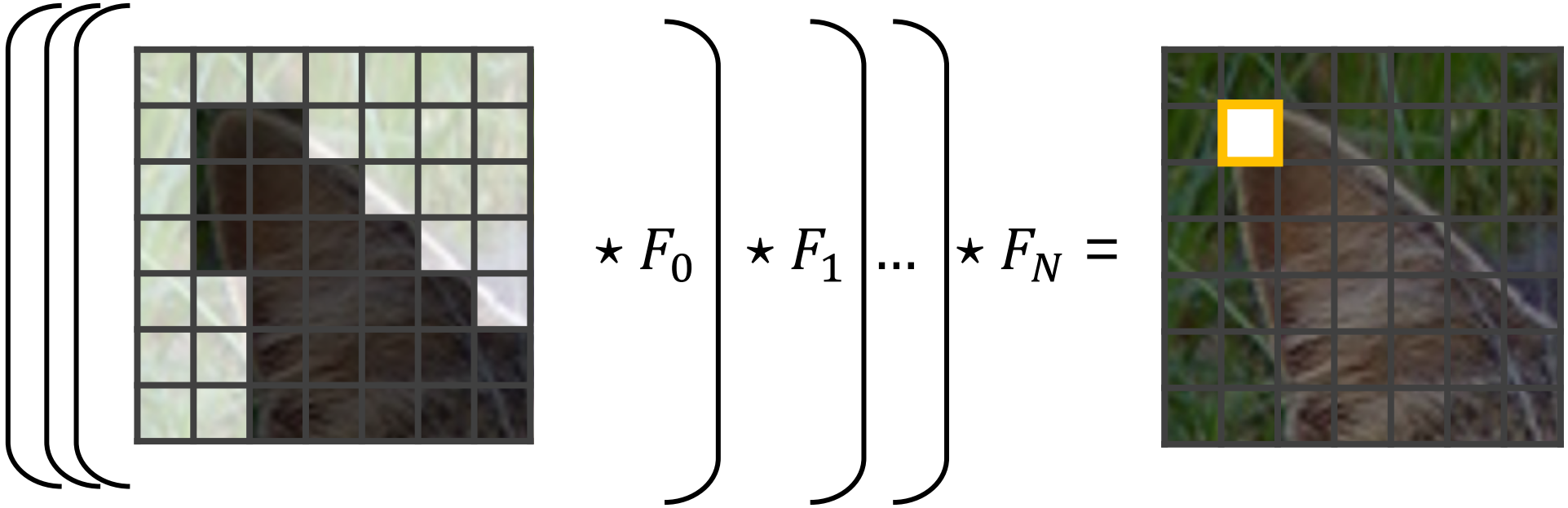
Example: Ear detector

Let us try to find ear-tip in the image manually.

Image

Ear-tip map

Filter cascade



Image

Correlations

Ear-tip map

1) Simple horizontal gradient

$Image * \boldsymbol{F_0}$

Correlation with a filter $F_0 = [-1, 1]$.

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

Image

$\star\, F_0 =$

-1	1

0						

Gradient

1) Simple horizontal gradient

$Image * \boldsymbol{F_0}$

Correlation with a filter $F_0 = [-1, 1]$.

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

Image

$\star\ F_0 =$

-1	1

0						
-1						

Gradient

1) Simple horizontal gradient

$Image * \boldsymbol{F_0}$

Correlation with a filter $F_0 = [-1, 1]$.

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

Image

$\star\, F_0 =$

-1	1

0	0	0	0	0	0	0
-1	0	1	0	0	0	0
-1	0	0	1	0	0	0
-1	0	0	0	1	0	0
0	-1	0	0	0	1	0
0	-1	0	0	0	0	0
0	-1	0	0	0	0	0

Gradient

1) Simple horizontal gradient

$Image * \boldsymbol{F_0}$

0	0	0	0	0	0	0
-1	0	1	0	0	0	0
-1	0	0	1	0	0	0
-1	0	0	0	1	0	0
0	-1	0	0	0	1	0
0	-1	0	0	0	0	0
0	-1	0	0	0	0	0

Gradient

2) Ear-tip detector

$(Image * F_0) * \boldsymbol{F_1}$

Build a 2nd level filter.

0	0	0	0	0	0	0
-1	0	1	0	0	0	0
-1	0	0	1	0	0	0
-1	0	0	0	1	0	0
0	-1	0	0	0	1	0
0	-1	0	0	0	0	0
0	-1	0	0	0	0	0

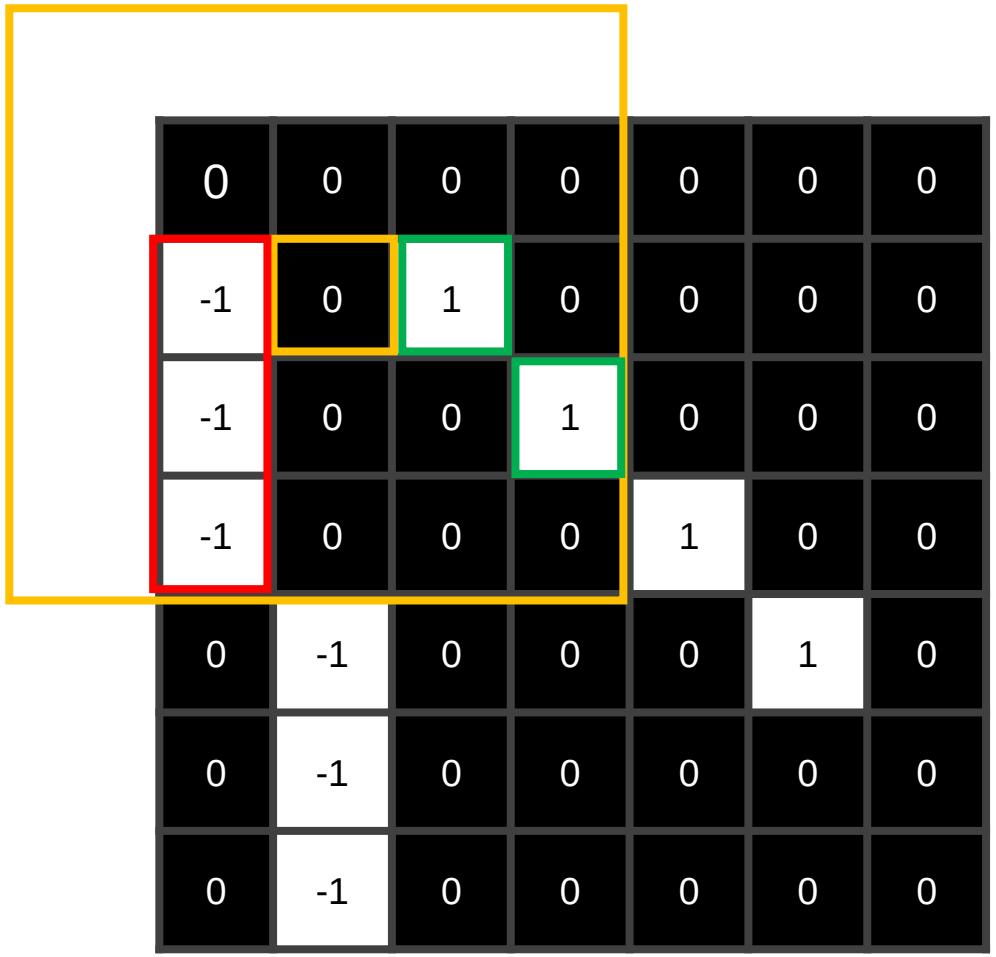
Gradient

$\star F_1 =$

2) Ear-tip detector

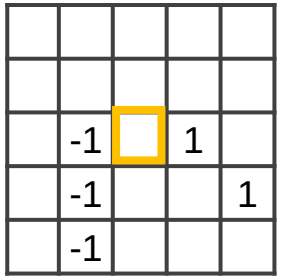
$(Image * F_0) * \boldsymbol{F_1}$

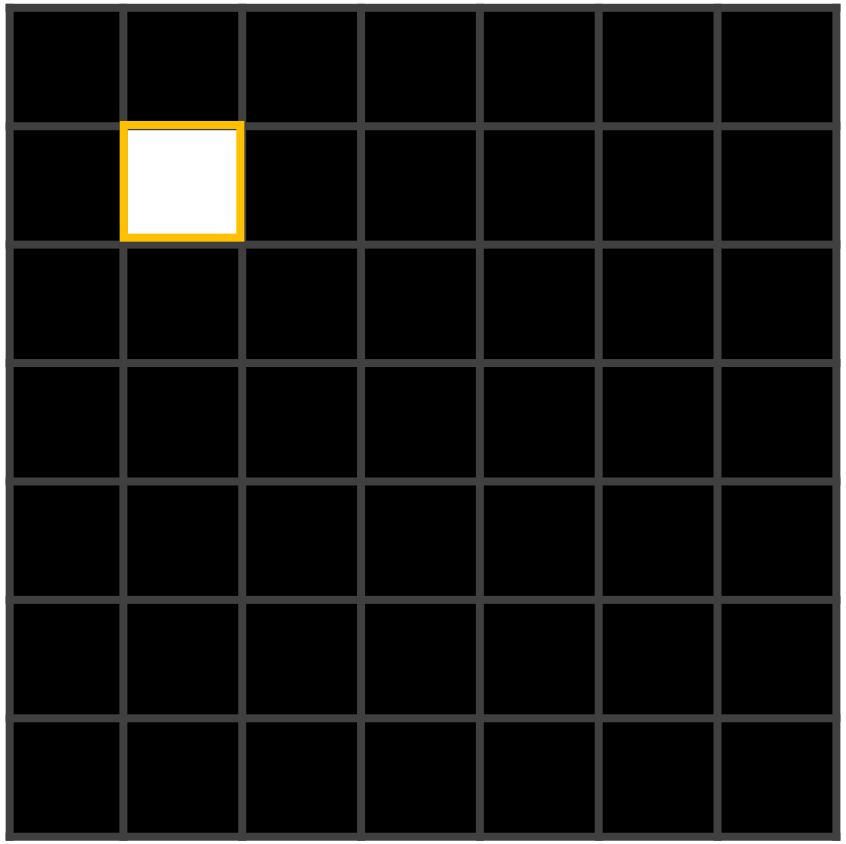
F_1



Gradient

$\star F_1 =$





Ear-tip map

2) Ear-tip detector

$(Image * F_0) * \boldsymbol{F_1}$

F_1

0	0	0	0	0	0	0
-1	0	1	0	0	0	0
-1	0	0	1	0	0	0
-1	0	0	0	1	0	0
0	-1	0	0	0	1	0
0	-1	0	0	0	0	0
0	-1	0	0	0	0	0

Gradient

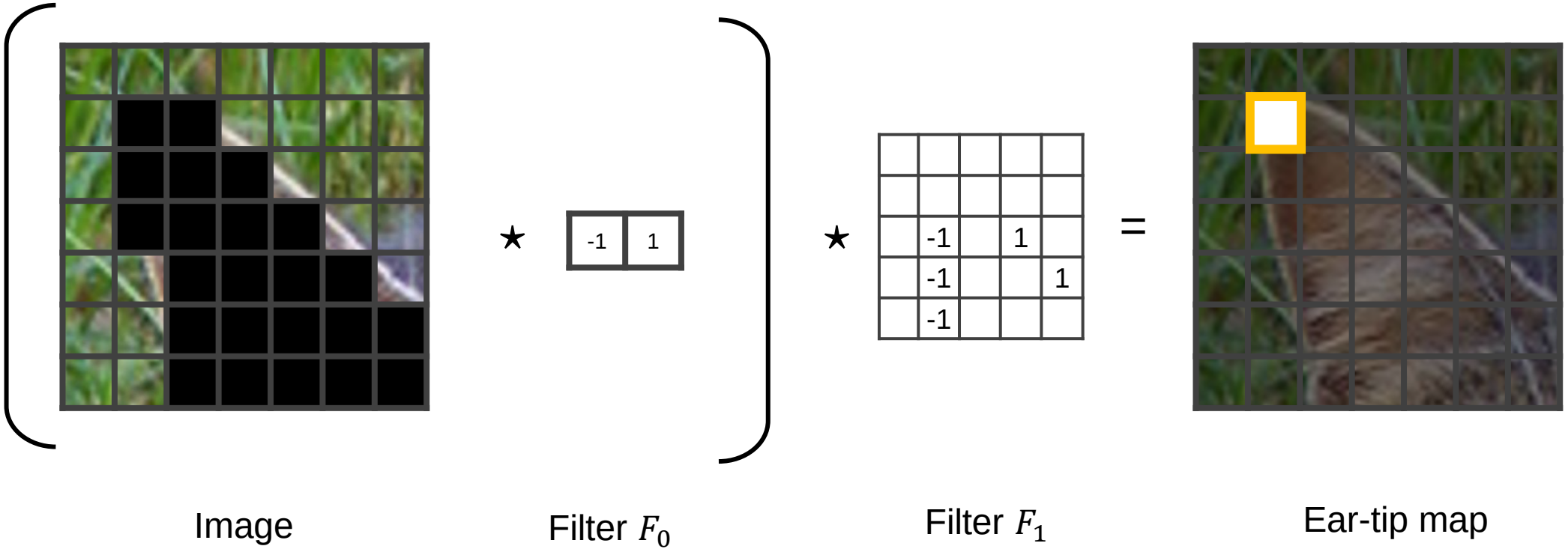
$\star F_1 =$

	-1		1	
	-1			1
	-1			

0	2	0	0	0	0	0
0	5	0	-1	0	0	0
0	2	3	0	-1	0	0
0	1	0	2	0	-1	0
0	0	3	0	1	0	-1
0	0	2	0	0	0	0
0	0	1	0	0	0	0

Ear-tip map

Final ear detector cascade



Results

Hand-crafted statistics will fit only some of the example images.

Credit: Andrew Wilkinson @ flickr

Credit: Chiara Coetzee @ flickr

Credit: kamanobe @ flickr

=> Instead of tuning each new case, we learn a universal descriptor from an image collection.

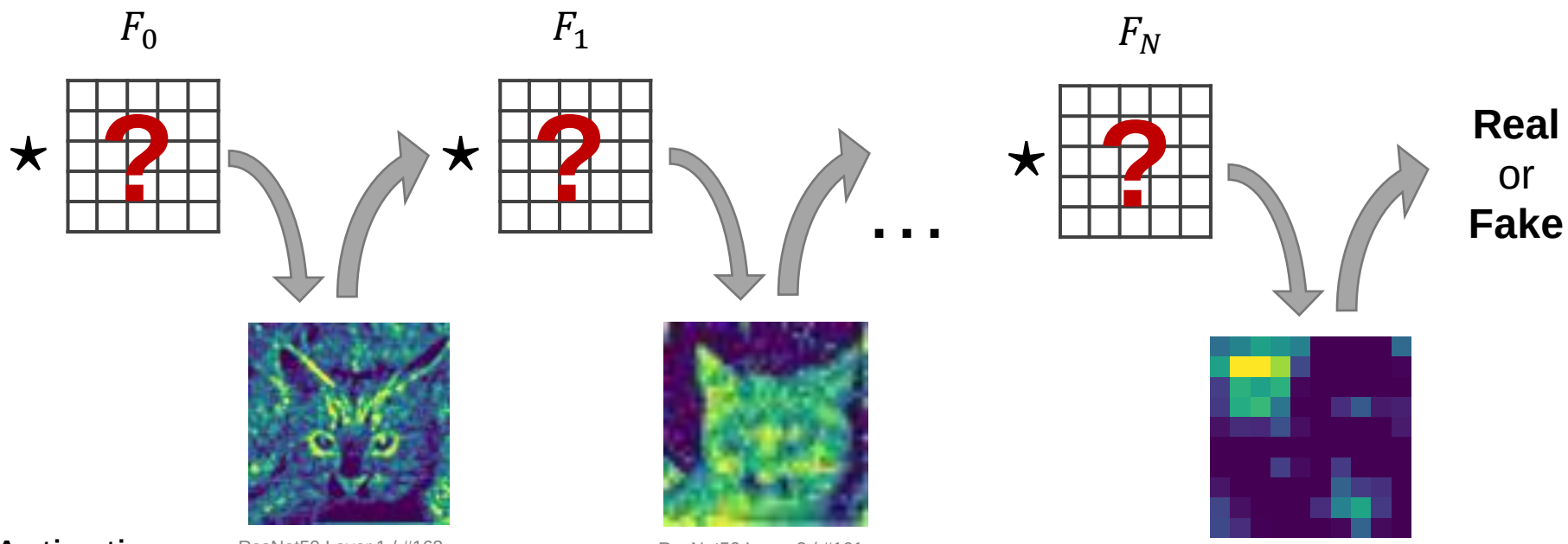
Learned statistics

Discriminator

Learn a sequence of high-dimensional filters from image collection.

Note: Nonlinearities omitted for simplicity.

$$Y_0 = \tau(I_0 \star F_0) \qquad Y_1 = \tau(Y_0 \star F_1) \qquad I_N = \tau(Y_{N-1} \star F_N)$$

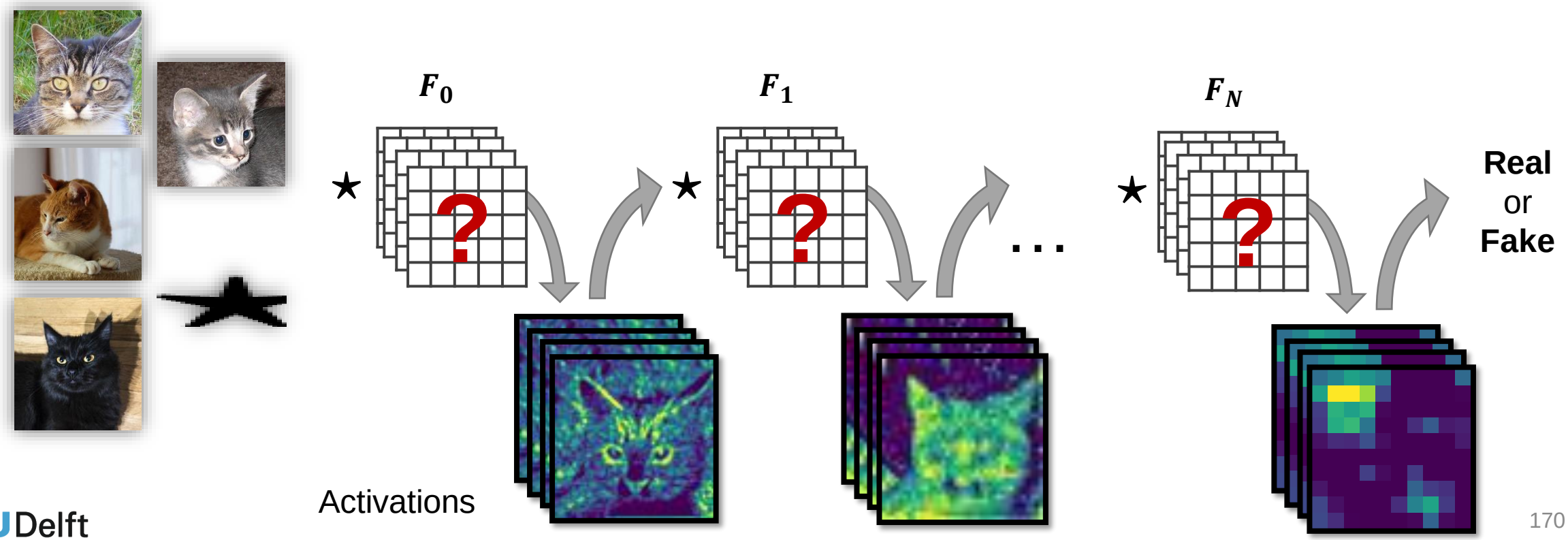


Learned statistics

Discriminator

Learn a sequence of high-dimensional filters from image collection.

$$\boldsymbol{Y_0} = \tau(I_0 \star \boldsymbol{F}_0) \qquad \boldsymbol{Y_1} = \tau(\boldsymbol{Y_0} \star \boldsymbol{F}_1) \qquad \boldsymbol{I_N} = \tau(\boldsymbol{Y_{N-1}} \star \boldsymbol{F}_N)$$



Learning Discriminator

Discriminator learns what makes an image “real”.

Credit: Osvaldo Bojanovich @ flickr

Real image from a collection

Actual response

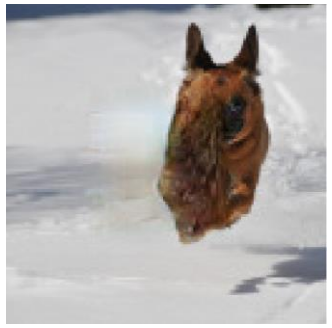
A desired response

Discriminator → **0.510** ⟷ **1**

Loss

“It is ok to have two eyes!”

[Iizuka 2017]



Inpainted image from the generator

Discriminator → **0.498** ⟷ **0**

Loss

“Dogs should have legs!”

Learning Discriminator

Discriminator learns what makes an image “real”.

Credit: Osvaldo Bojanovich @ flickr

Real image from a collection

Actual response

A desired response

Discriminator

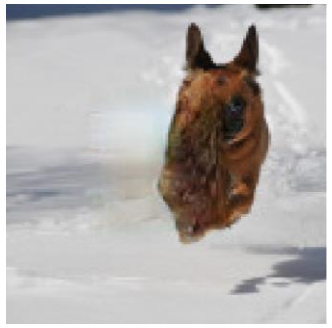
0.854 ↑

1

Loss

“It is ok to have two eyes!”

[Iizuka 2017]



Inpainted image from the generator

Discriminator

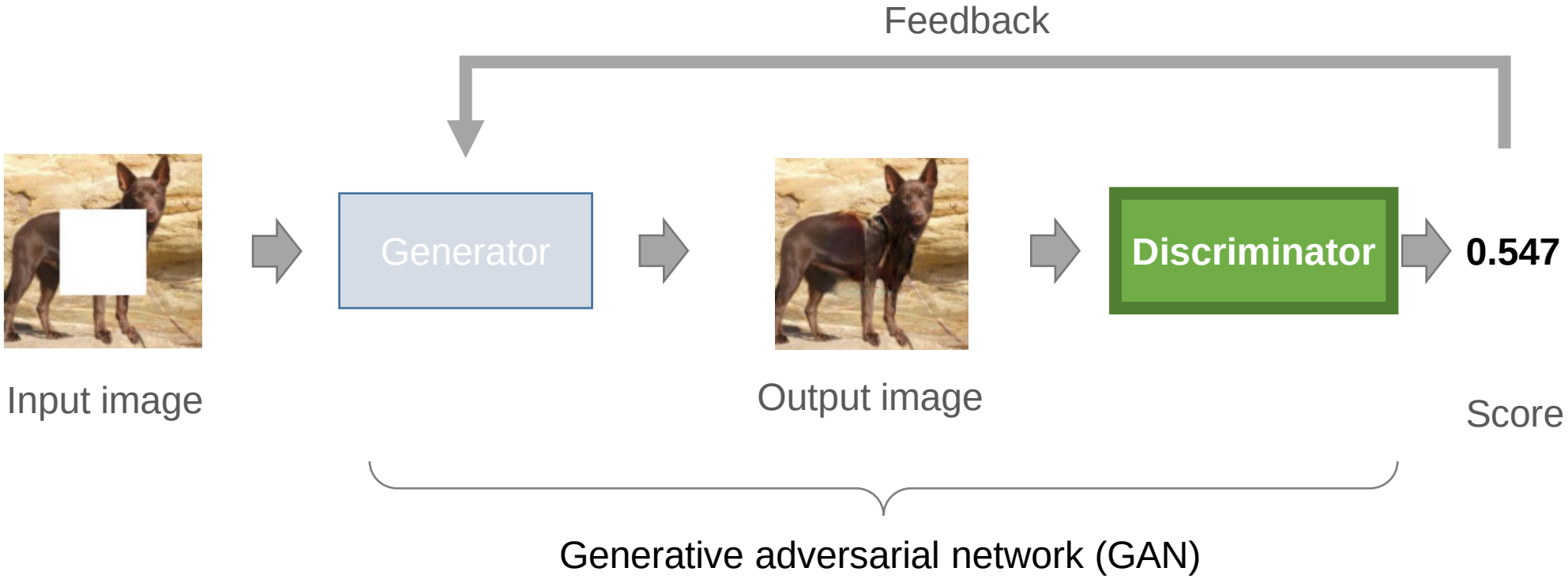
0.214 ↓

0

Loss

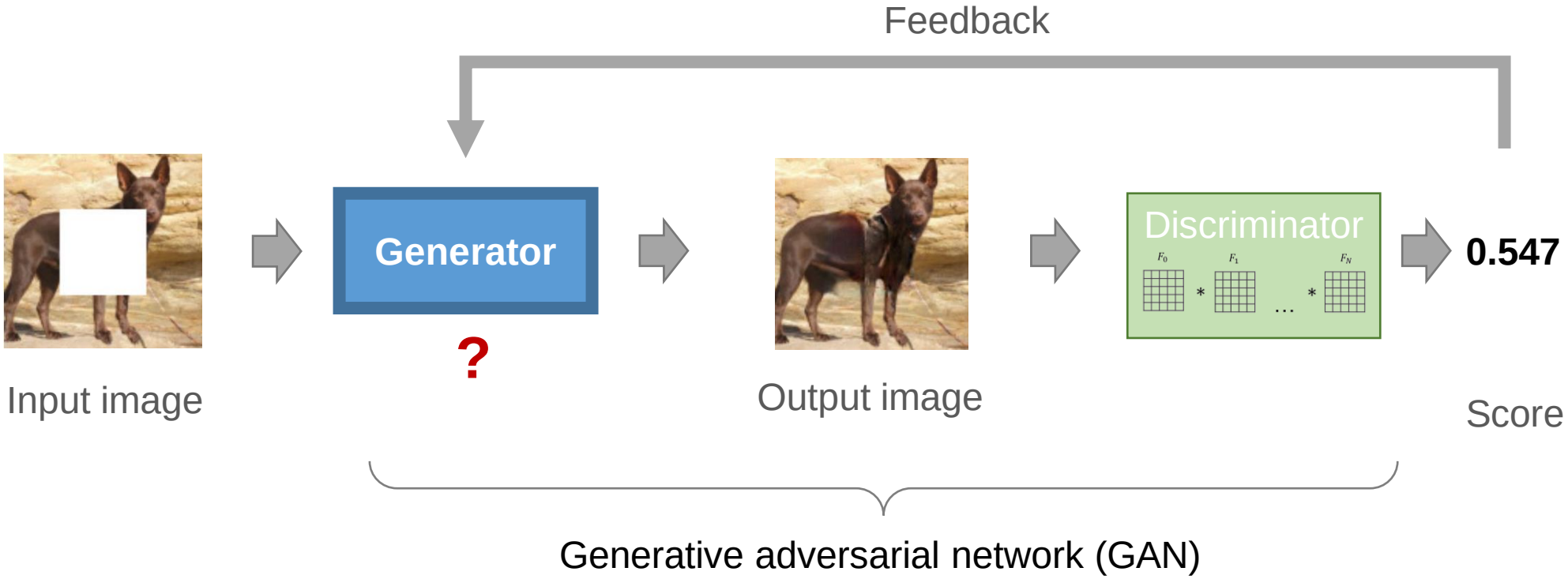
“Dogs should have legs!”

Neural inpainting using GAN



Iizuka, Satoshi, Edgar Simo-Serra, and Hiroshi Ishikawa. "Globally and locally consistent image completion." . SIGGRAPH 2017.

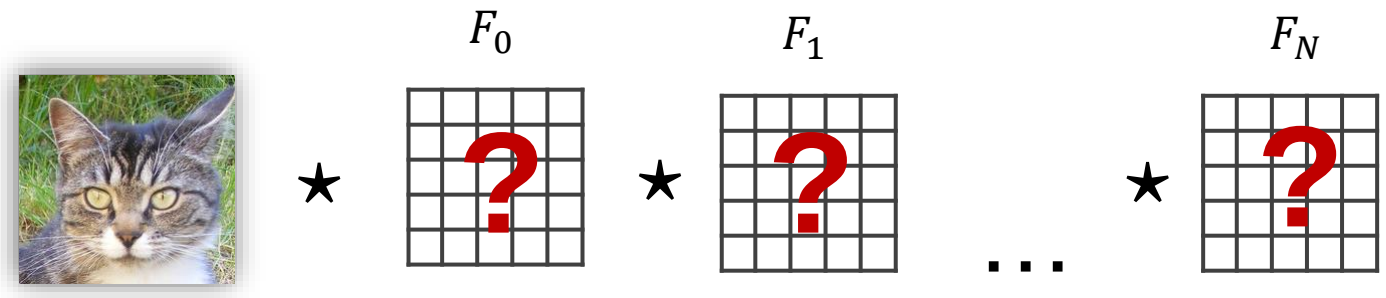
Neural inpainting using GAN



Iizuka, Satoshi, Edgar Simo-Serra, and Hiroshi Ishikawa. "Globally and locally consistent image completion." . SIGGRAPH 2017.

What does the network learn?

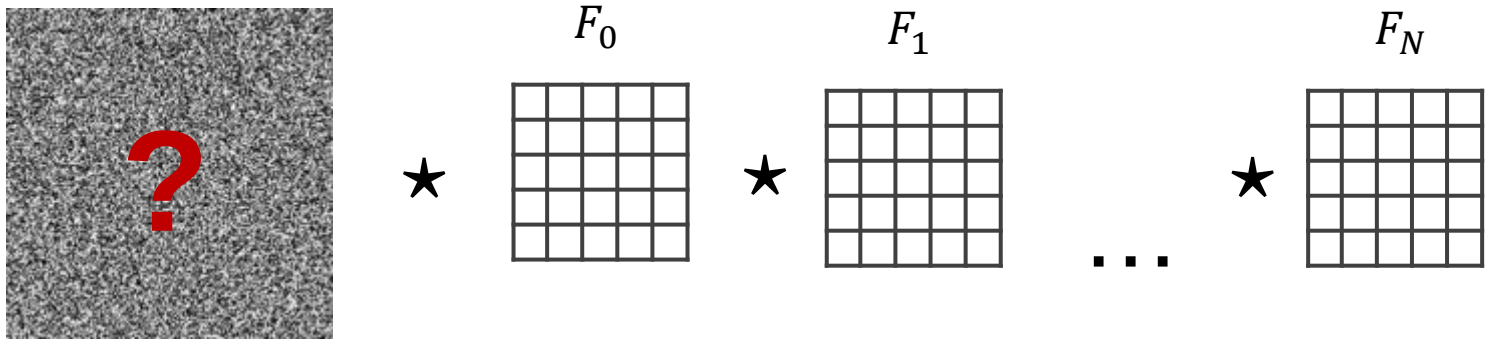
We can run the correlations “backwards” to find “optimal” input.



$$Y = I \star F$$

What does the network learn?

We can run the correlations “backwards” to find “optimal” input.



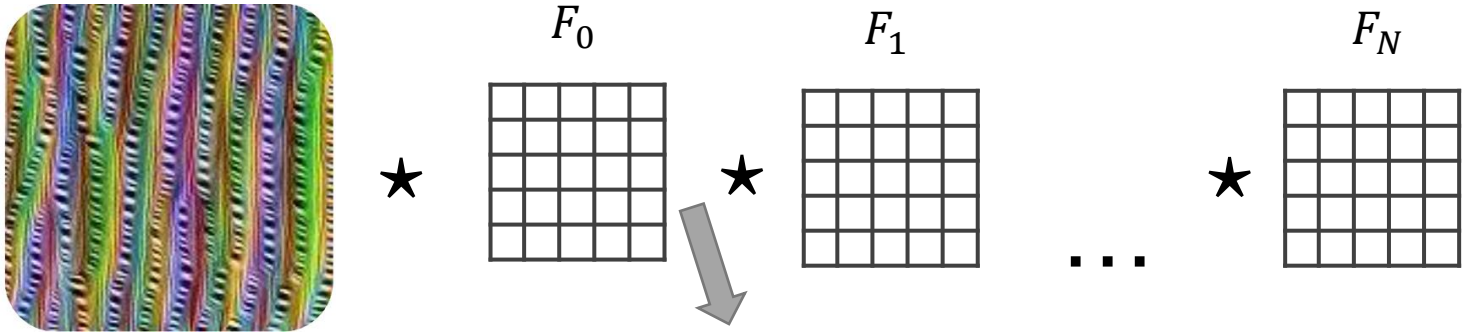
Unknown input

$$Y = I \star F$$

$$I = \underset{I}{\operatorname{argmax}} \, |I \star F|$$

What does the network learn?

We can run the correlations “backwards” to find “optimal” input.



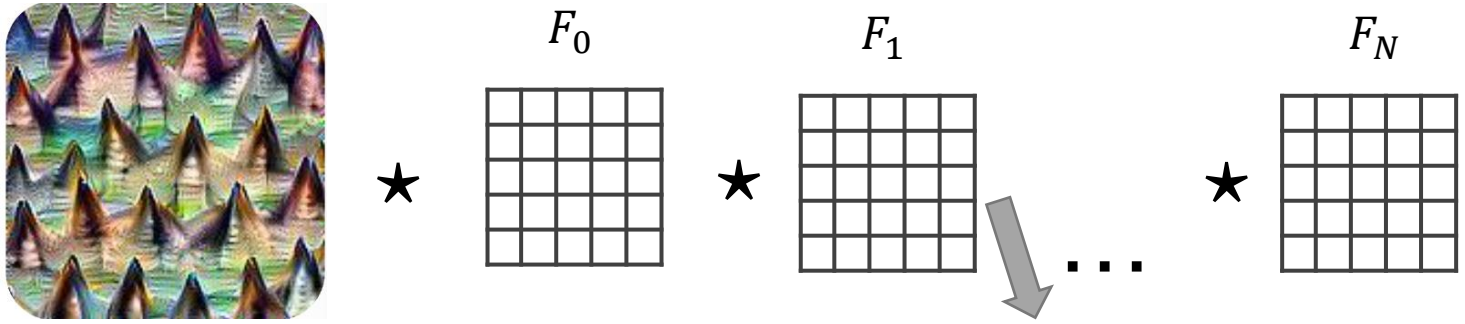
$$Y = I \star F$$

$$I = \underset{I}{\operatorname{argmax}} \, |I \star F|$$

Hohman, Fred, et al. "Summit: Scaling deep learning interpretability by visualizing activation and attribution summarizations." *IEEE TVCG (2019)*

What does the network learn?

We can run the correlations “backwards” to find “optimal” input.



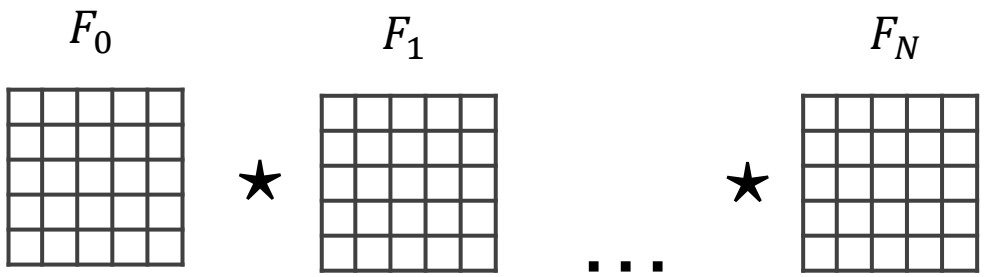
$$Y = I \star F$$

$$I = \underset{I}{\operatorname{argmax}} \, |I \star F|$$

Hohman, Fred, et al. "Summit: Scaling deep learning interpretability by visualizing activation and attribution summarizations." *IEEE TVCG (2019)*

What does the network learn?

The filters extract progressively complex image statistics.



$$Y = I \star F$$

$$I = \underset{I}{\operatorname{argmax}} \, |I \star F|$$

Correlation

Decreases the input dimensions.

0	1	1
2	3	5
3	2	-1

$\star$

1	0
2	-1

$=$ **?**

Input: 3 x 3 **Filter**: 2 x 2

Correlation

Decreases the input dimensions.

$w_{in} = 3$ $\quad$ $k = 2$ $\quad$ $w_{out} = 2$

$$\begin{bmatrix} 0 & 1 & 1 \\ 2 & 3 & 5 \\ 3 & 2 & -1 \end{bmatrix} \star \begin{bmatrix} 1 & 0 \\ 2 & -1 \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 6 & 8 \end{bmatrix}$$

Input: 3 x 3 $\quad$ > $\quad$ **Output**: 2 x 2

$$w_{out} = w_{in} - k + 1$$

$$w_{out} = \mathbf{3} - 2 + 1 = \mathbf{2}$$

Transposed correlation

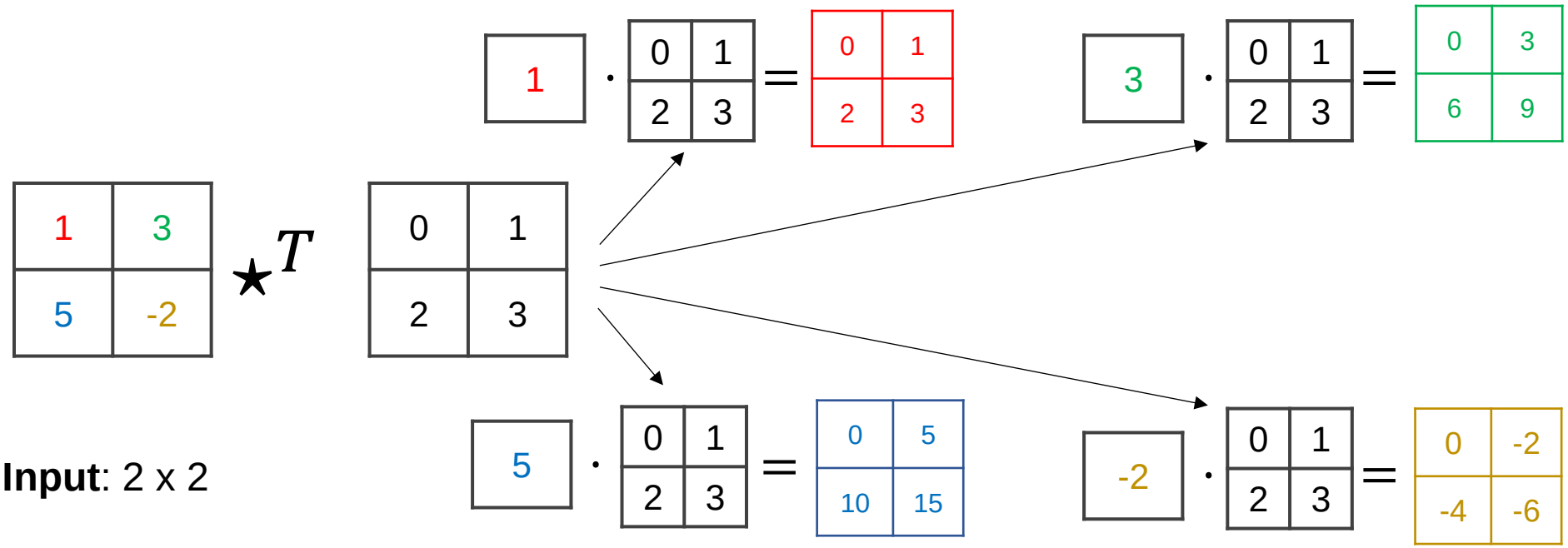
Increases the input dimensions.

$$\begin{bmatrix} 1 & 3 \\ 5 & -2 \end{bmatrix} \star^T \begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} = \; ?$$

Input: 2 x 2 **Filter**: 2 x 2

Transposed correlation

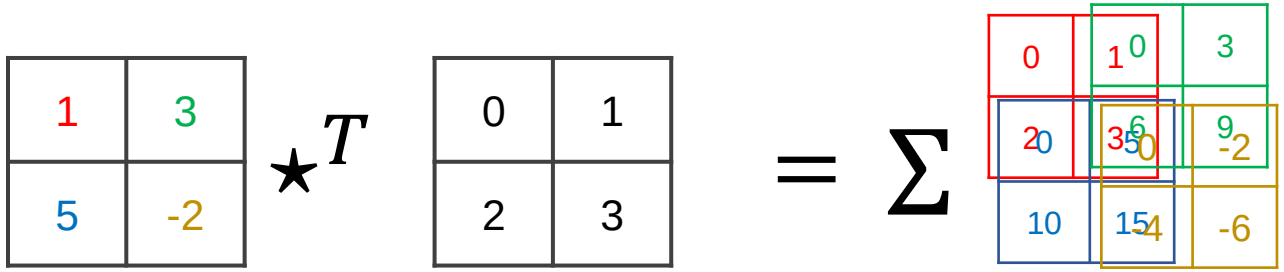
Increases the input dimensions.



Input: 2 x 2

Transposed correlation

Increases the input dimensions.



Transposed correlation

Increases the input dimensions.

$w_{in} = 2$

1	3
5	-2

$\star^T$

$k = 2$

0	1
2	3

$=$

$w_{out} = 3$

0	1	3
2	14	7
10	11	-6

Input: 2 x 2 **Filter**: 2 x 2 **<** **Output**: 3 x 3

$$w_{out} = w_{in} + k - 1$$

$$w_{out} = \mathbf{2} + 2 - 1 = \mathbf{3}$$

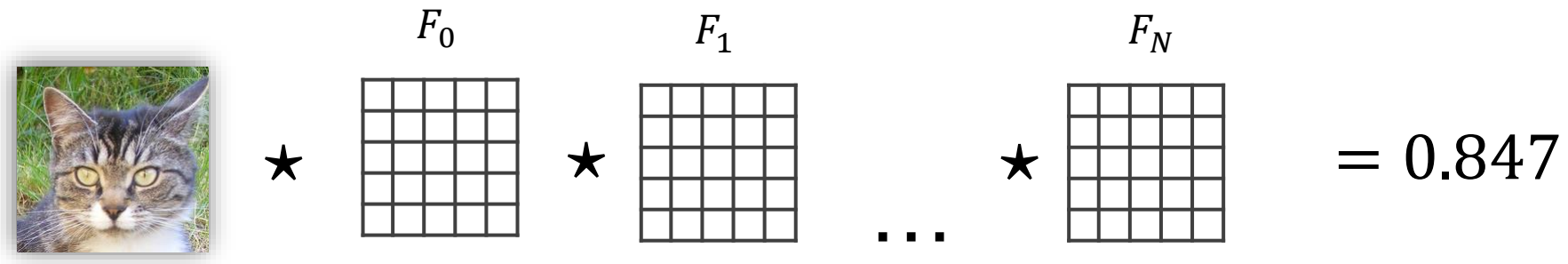
	Correlation $\star$	Transposed correlation $\star^T$
Simple case: (no padding, stride = 1, dilation = 1)	$w_{out} = w_{in} - k + 1$ Input size >= Output size	$w_{out} = w_{in} + k - 1$ Input size <= Output size
General case: (**not** in exam)	$W_{out} = \left\lfloor \frac{W_{in} + 2 \times \text{padding}[1] - \text{dilation}[1] \times (\text{kernel_size}[1] - 1) - 1}{\text{stride}[1]} + 1 \right\rfloor$	$W_{out} = (W_{in} - 1) \times \text{stride}[1] - 2 \times \text{padding}[1] + \text{dilation}[1] \times (\text{kernel_size}[1] - 1) + \text{output_padding}[1] + 1$

https://pytorch.org/docs/1.8.0/generated/torch.nn.Conv2d.html?highlight=conv2d#torch.nn.Conv2d

Image discriminator

Discriminator

A cascade of correlations.



Image

High-dimensional input

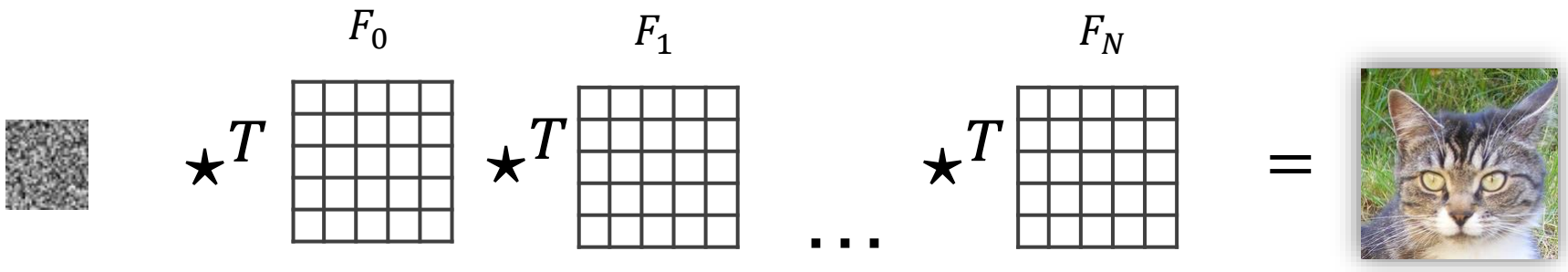
Score

Low-dimensional output

Image generator

Generator

A cascade of **transposed** correlations.



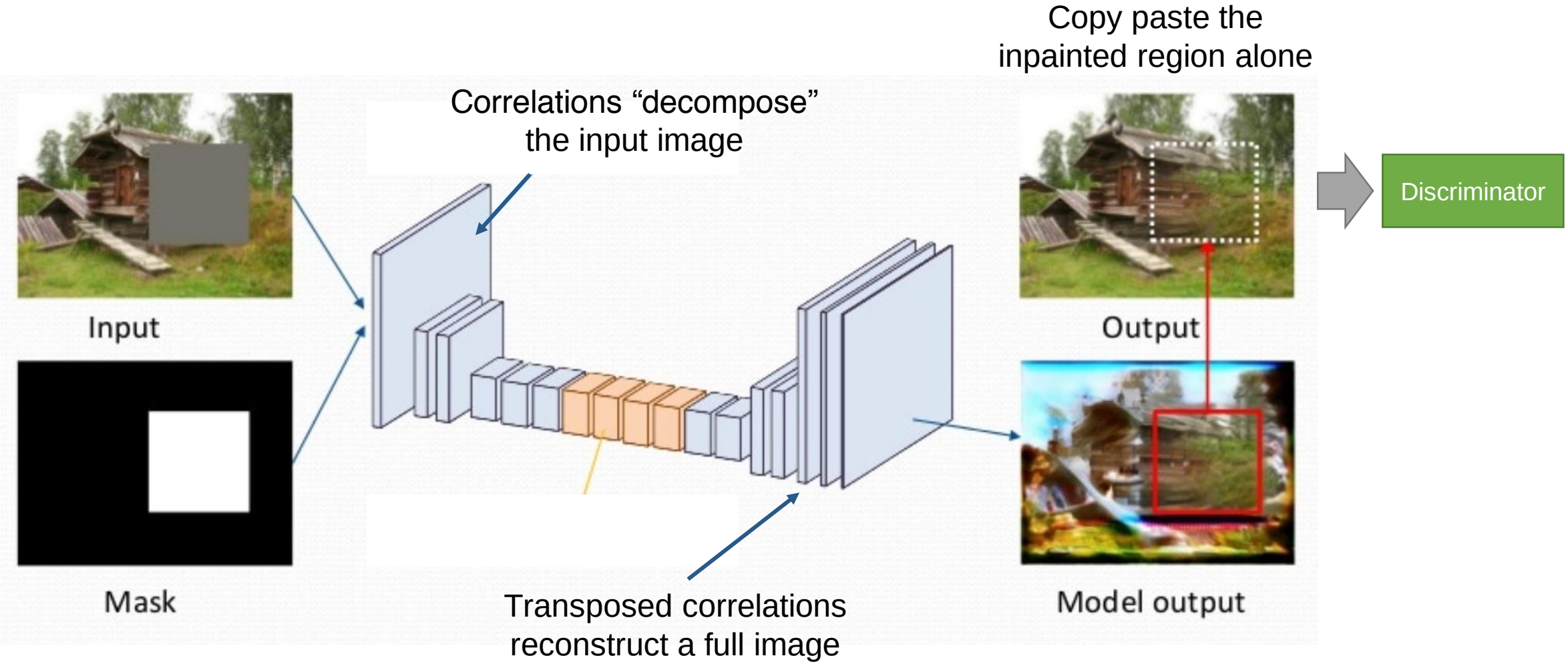
Random input

Low-dimensional input

Image

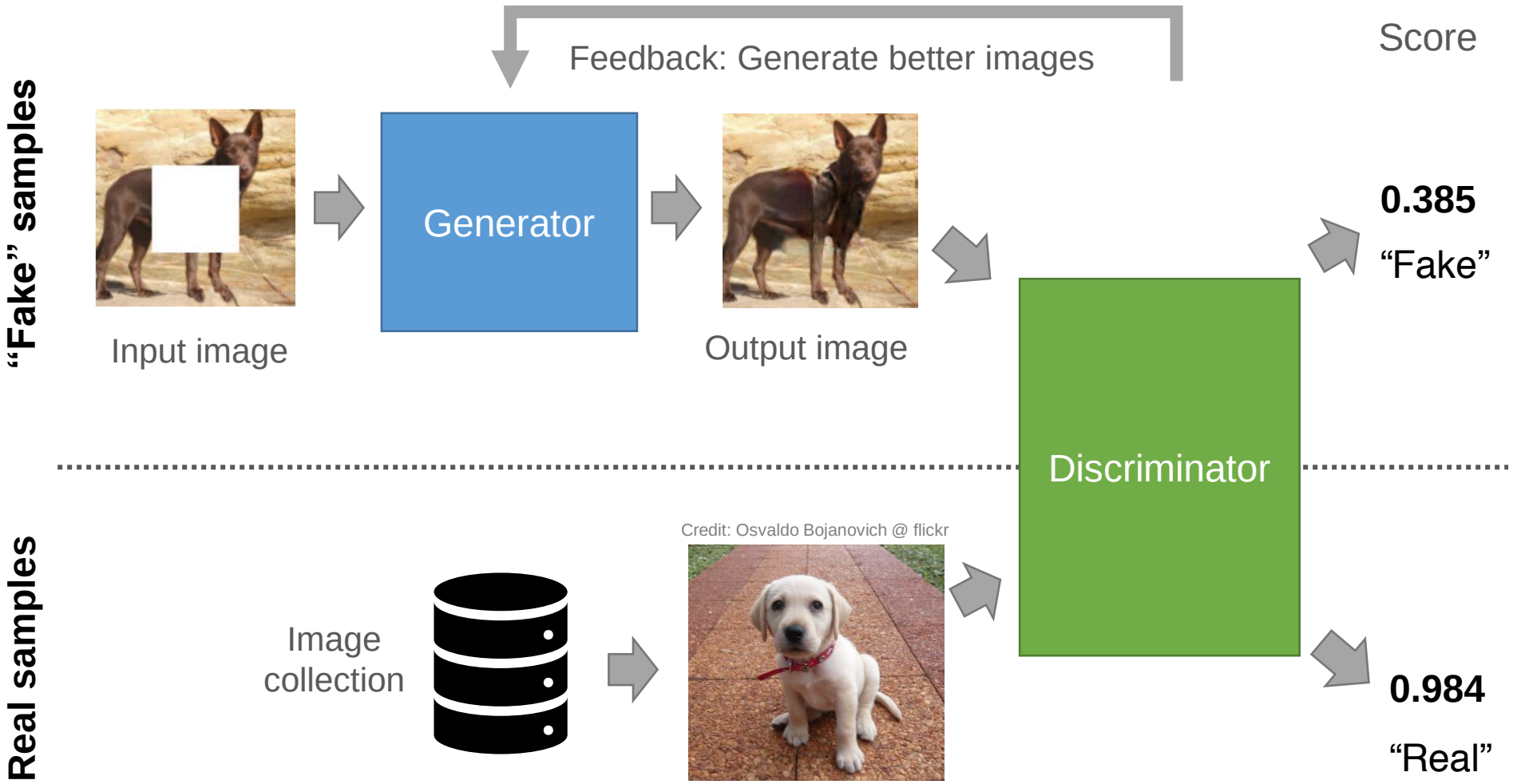
High-dimensional output

Generator example



Iizuka, Satoshi, Edgar Simo-Serra, and Hiroshi Ishikawa. "Globally and locally consistent image completion." . SIGGRAPH 2017.

Training full GAN



Adversarial Goals

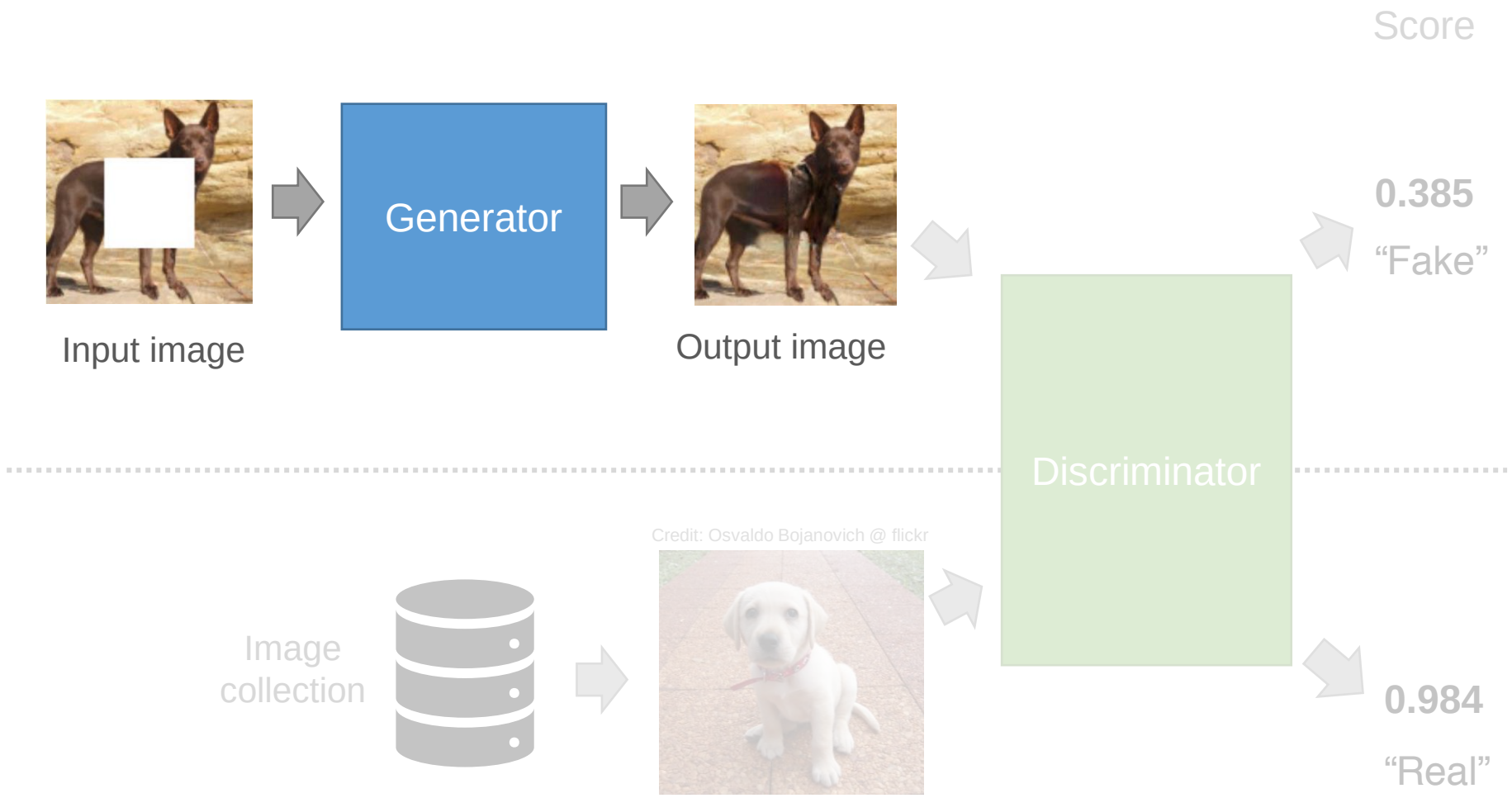
Generator: Make so good "**Fake**" images that **D.** think they are "**Real**".

versus

Discriminator: Correctly recognize **Real** vs. **Fake**.

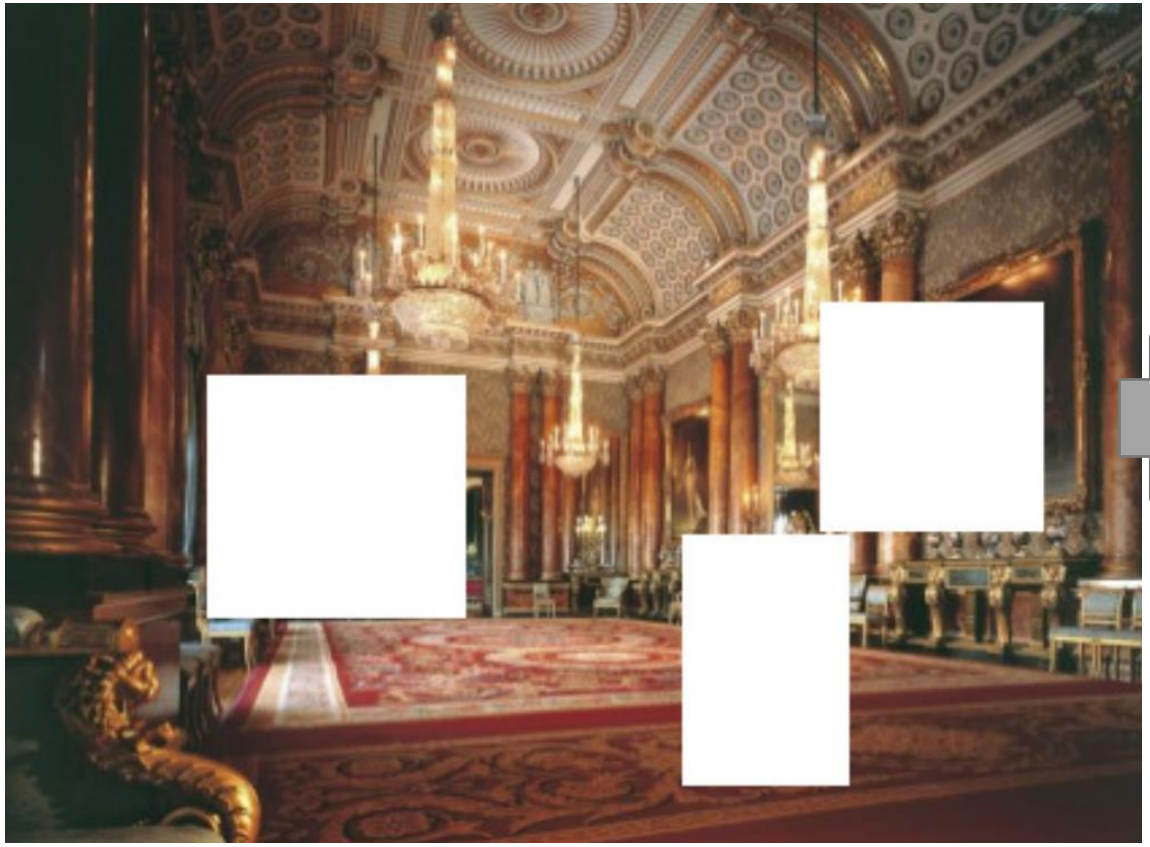
Iizuka, Satoshi, Edgar Simo-Serra, and Hiroshi Ishikawa. "Globally and locally consistent image completion." . SIGGRAPH 2017.

Inference



Iizuka, Satoshi, Edgar Simo-Serra, and Hiroshi Ishikawa. "Globally and locally consistent image completion." . SIGGRAPH 2017.

Example

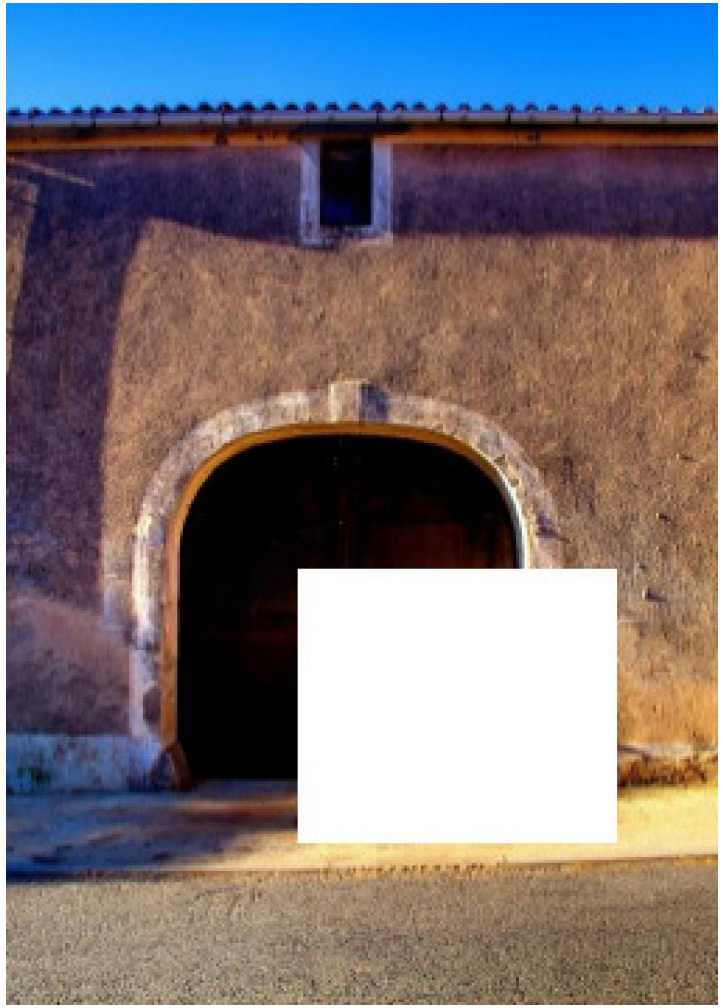


Input

Inpainted image

Iizuka, Satoshi, Edgar Simo-Serra, and Hiroshi Ishikawa. "Globally and locally consistent image completion." . SIGGRAPH 2017.

Comparison to Patch Match

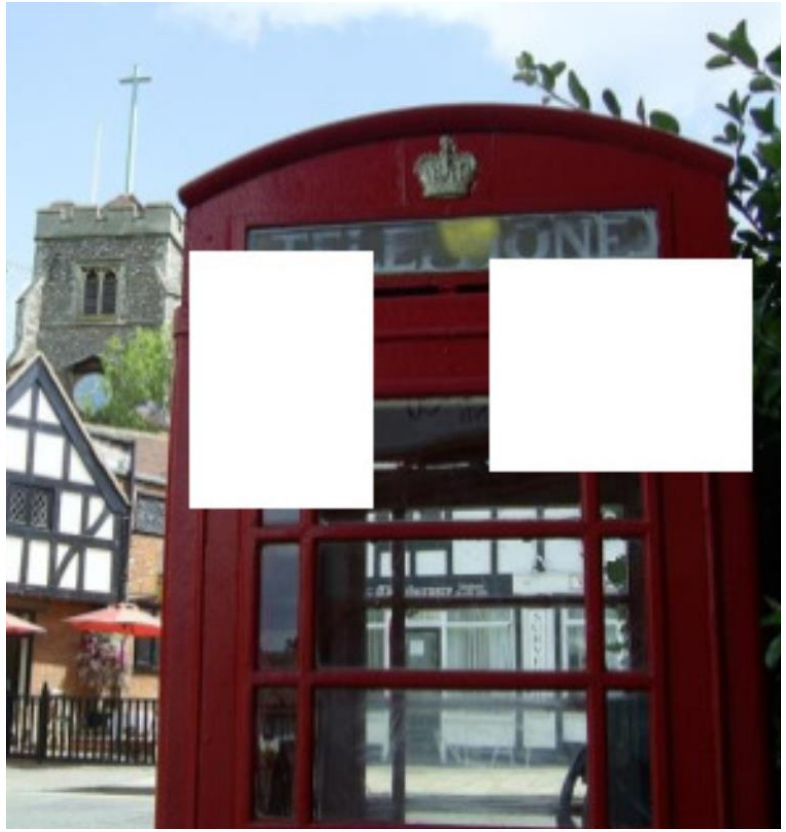


Input

Patch Match

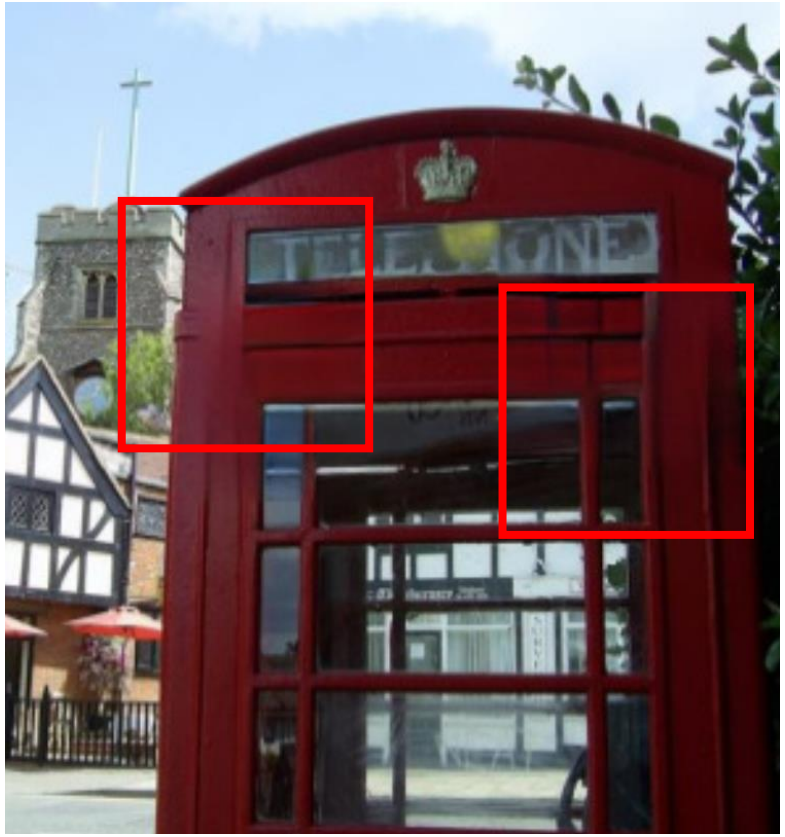
Neural inpainting

Comparison to Patch Match



Input

Patch Match



Neural inpainting

Iizuka, Satoshi, Edgar Simo-Serra, and Hiroshi Ishikawa. "Globally and locally consistent image completion." . SIGGRAPH 2017.

Comparison to Patch Match

Input

Patch Match

Neural inpainting

Iizuka, Satoshi, Edgar Simo-Serra, and Hiroshi Ishikawa. "Globally and locally consistent image completion." . SIGGRAPH 2017.

Comparison to Collection sampling

Input

Collection sampling

Neural inpainting

Iizuka, Satoshi, Edgar Simo-Serra, and Hiroshi Ishikawa. "Globally and locally consistent image completion." . SIGGRAPH 2017.

Comparison to Collection sampling

Input

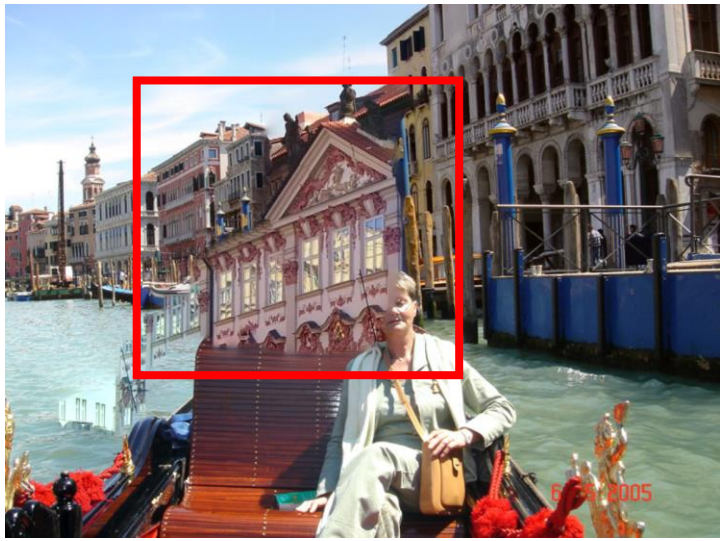
Collection sampling

Neural inpainting

Iizuka, Satoshi, Edgar Simo-Serra, and Hiroshi Ishikawa. "Globally and locally consistent image completion." . SIGGRAPH 2017.

Comparison to Collection sampling

Input



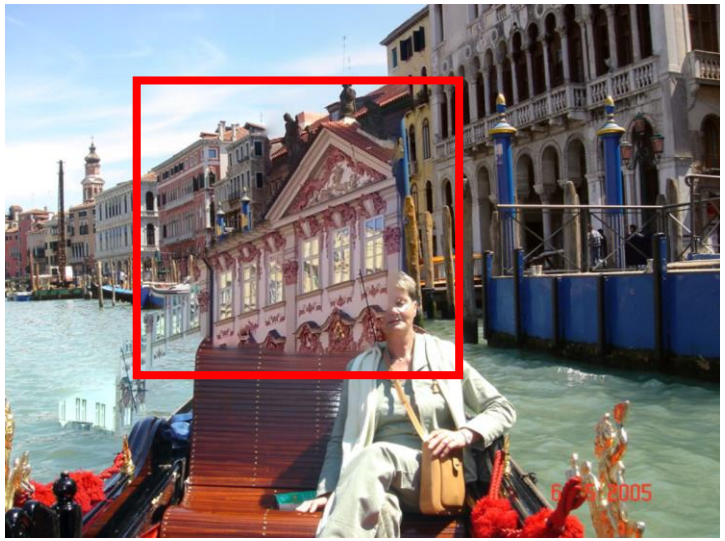
Collection sampling

Neural inpainting

Iizuka, Satoshi, Edgar Simo-Serra, and Hiroshi Ishikawa. "Globally and locally consistent image completion." . SIGGRAPH 2017.

Comparison to Collection sampling and PM

Input



Collection sampling

Neural inpainting

A modified Patch Match

Yuan Ting Hseih
https://github.com/YuanTingHsieh/Image_Completion

Neural inpainting using GAN

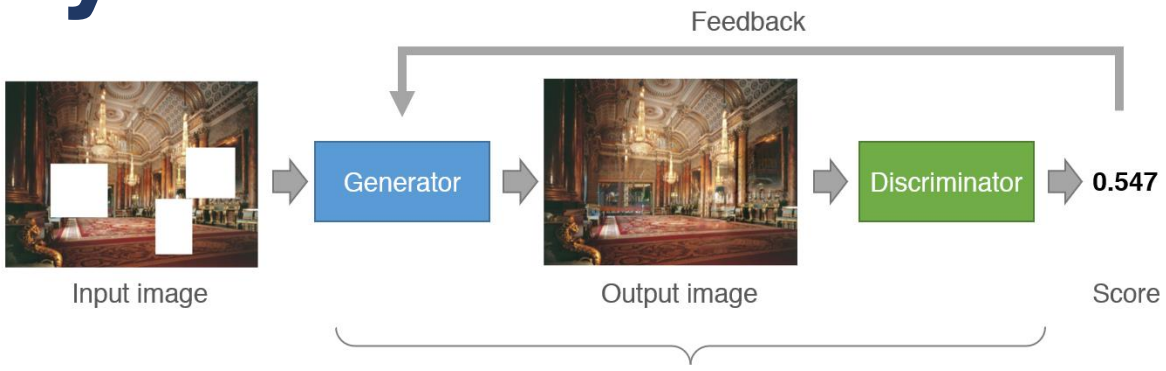
Good

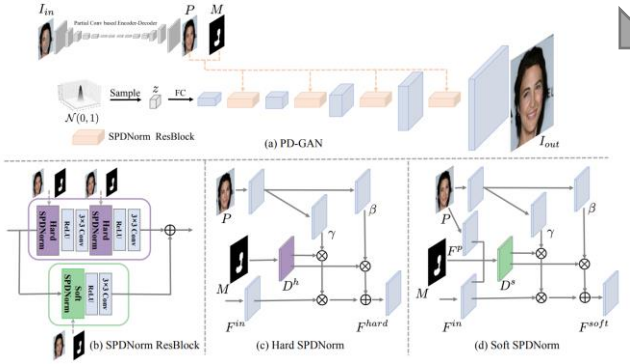
- Can create new content.
- Simple, modular, easy to experiment with.

Bad

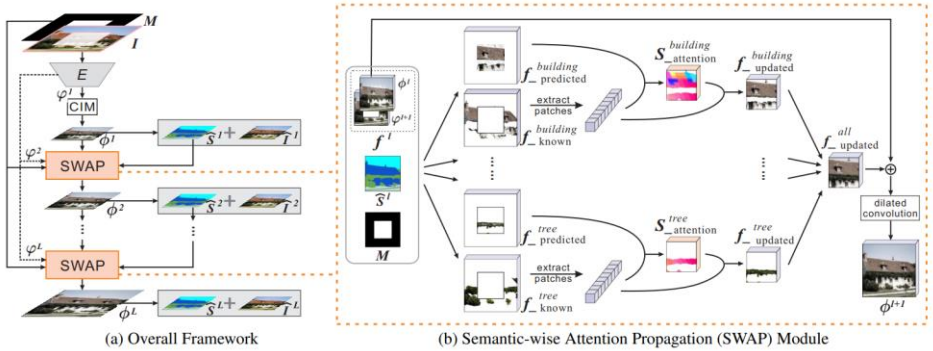
- Requires large training data.
- The image quality may vary.
- Failure cases are more unpredictable.
- Large holes are difficult.

Modularity

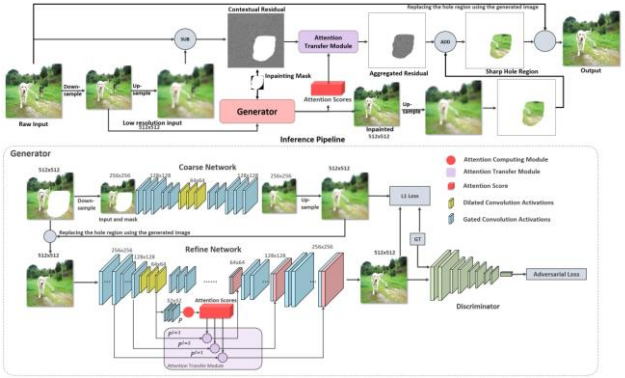




Liu, Hongyu, et al. "PD-GAN: Probabilistic Diverse GAN for Image Inpainting." CVPR 2021.



Yi, Zili, et al. "Contextual residual aggregation for ultra high-resolution image inpainting." CVPR 2020.



Liao, Liang, et al. "Image Inpainting Guided by Coherence Priors of Semantics and Textures." CVPR 2021.

Keeps improving

Input

Neural inpainting by [Iizuka 2017]

Neural inpainting by [Yu 2019]

Yu, Jiahui, et al. "Free-*form image inpainting with gated convolution." ICCV 2019.*

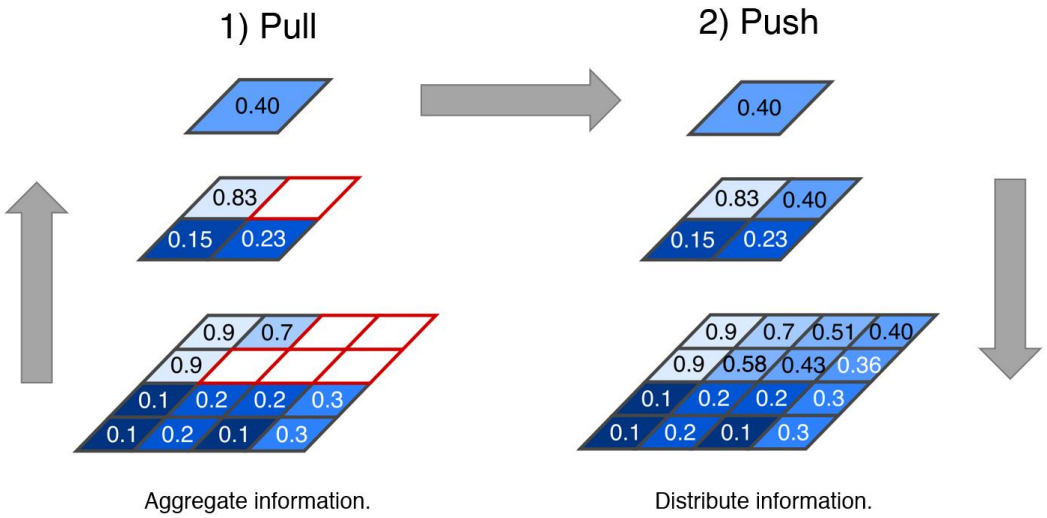
Research continues…

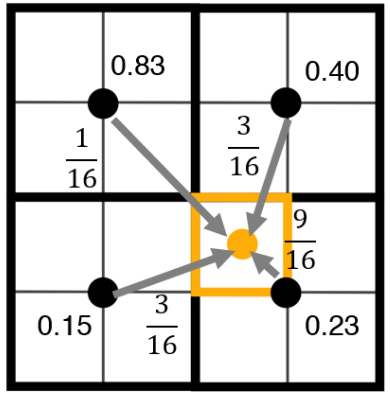
LaMa

Suvorov, Roman, et al. "Resolution-robust large mask inpainting with fourier convolutions." Proceedings of the IEEE/CVF winter conference on applications of computer vision. 2022.

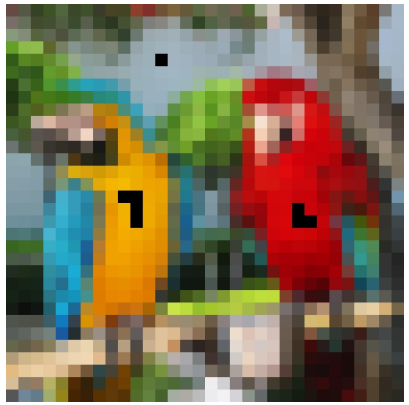
Summary

1. Pull-Push



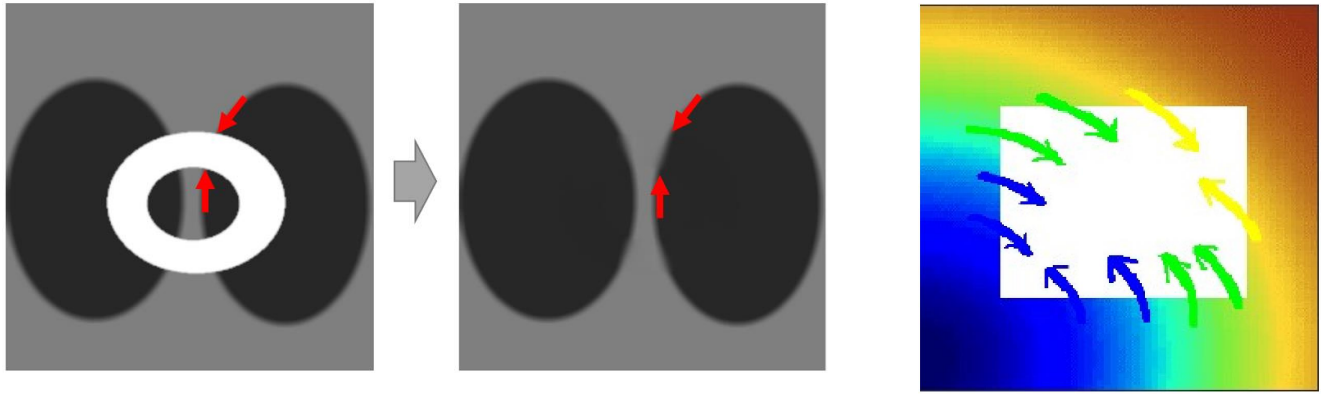


Bilinear interpolation



Summary

1. Pull-Push
2. **PDE Inpainting**



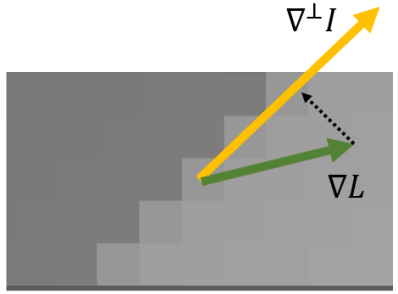


Image update

Inner product => projection

$$U = \underline{\nabla L} \cdot \underline{\nabla^{\perp} I} = 0$$

Change of information

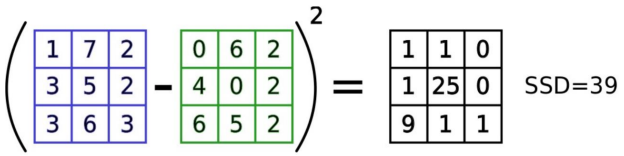
Propagation direction (Isophote)

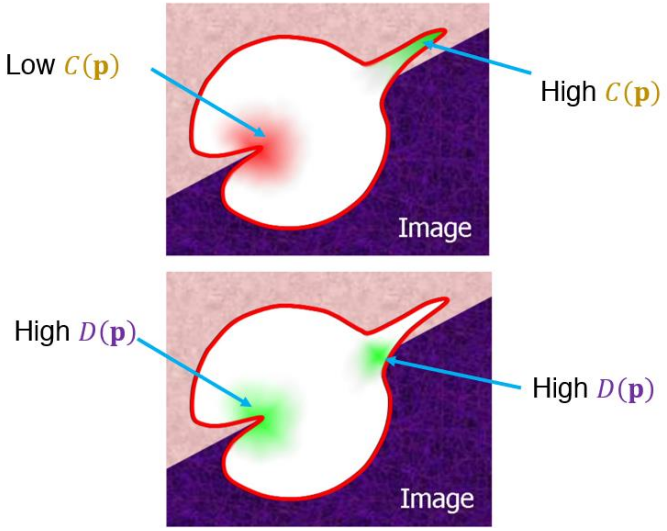
Information propagation along the isophotes

Summary

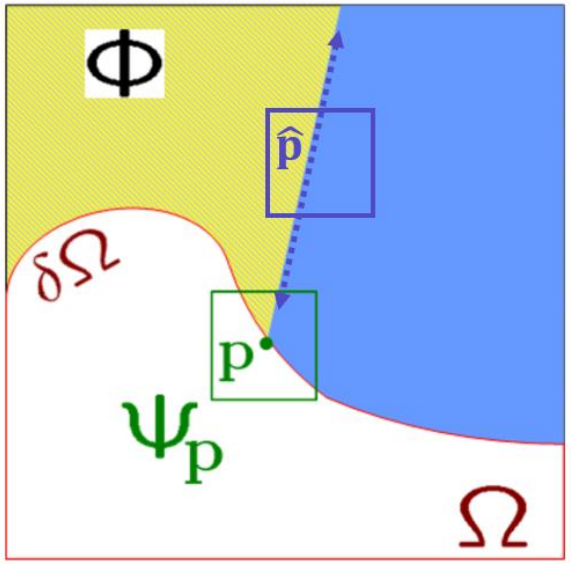
1. Pull-Push
2. PDE Inpainting
3. **Exemplar Inpainting**

$$P(\mathbf{p}) = C(\mathbf{p}) \cdot D(\mathbf{p}), \forall \mathbf{p} \in \delta\Omega$$





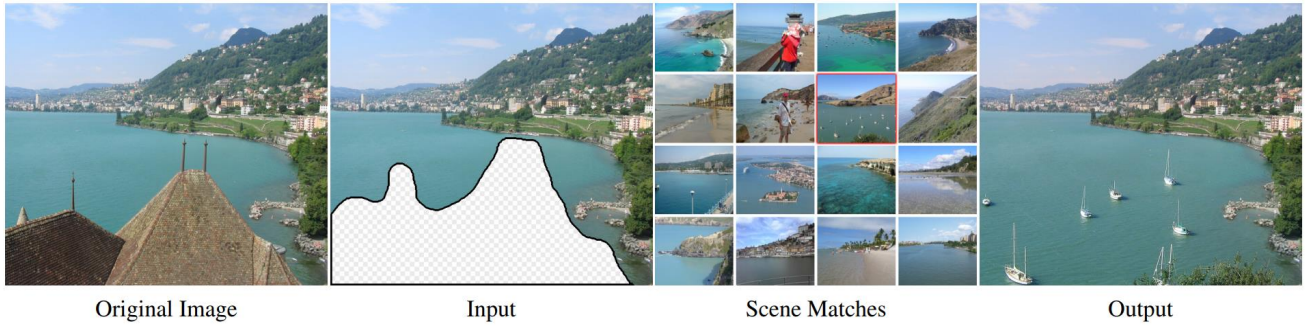
Ordering patches by priority



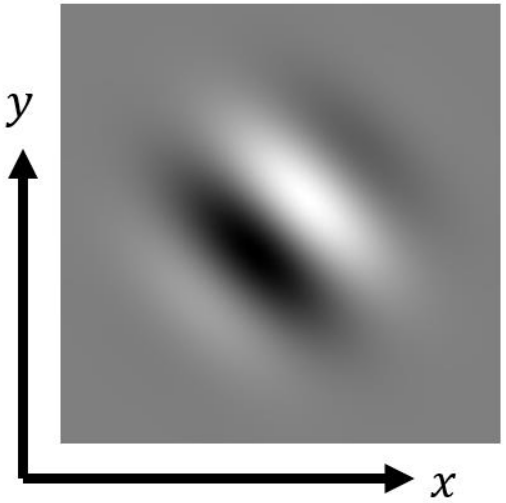
Selecting the most similar patch

Summary

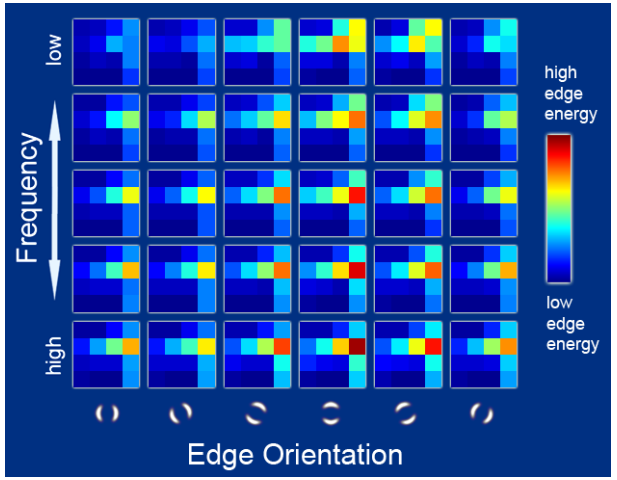
1. Pull-Push
2. PDE Inpainting
3. Exemplar Inpainting
4. **Collection Sampling**



Collection sampling



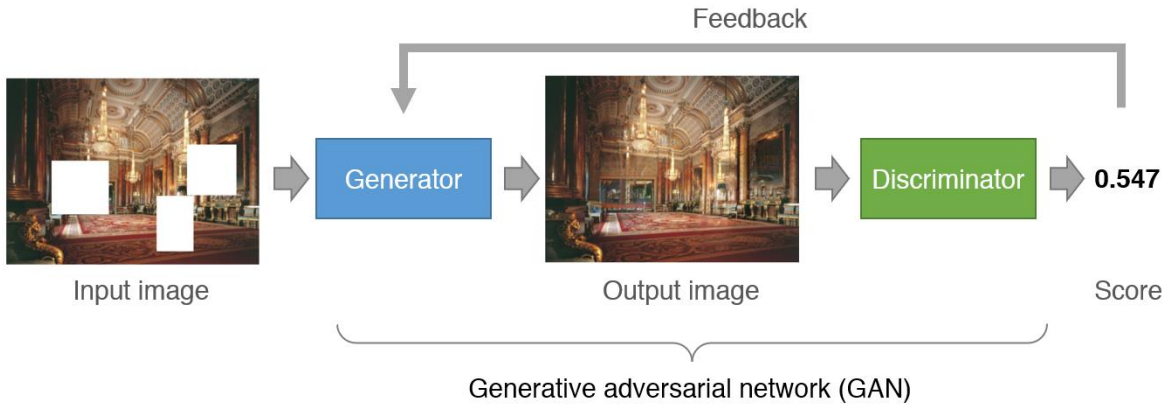
Gabor filter

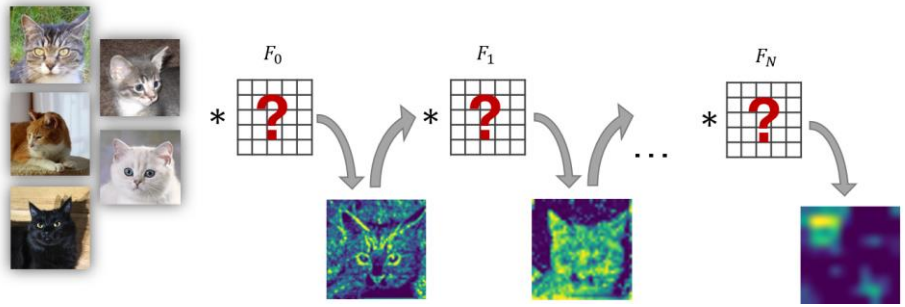


GIST descriptor

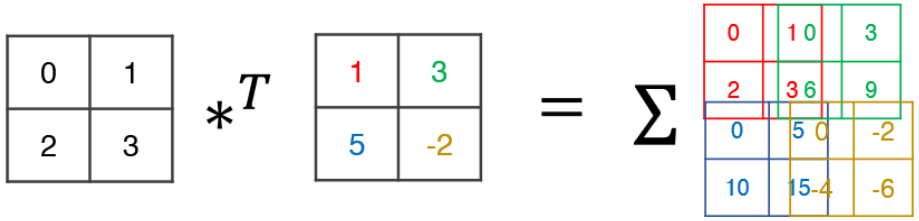
Summary

1. Pull-Push
2. PDE Inpainting
3. Exemplar Inpainting
4. Collection Sampling
5. **Neural Inpainting**





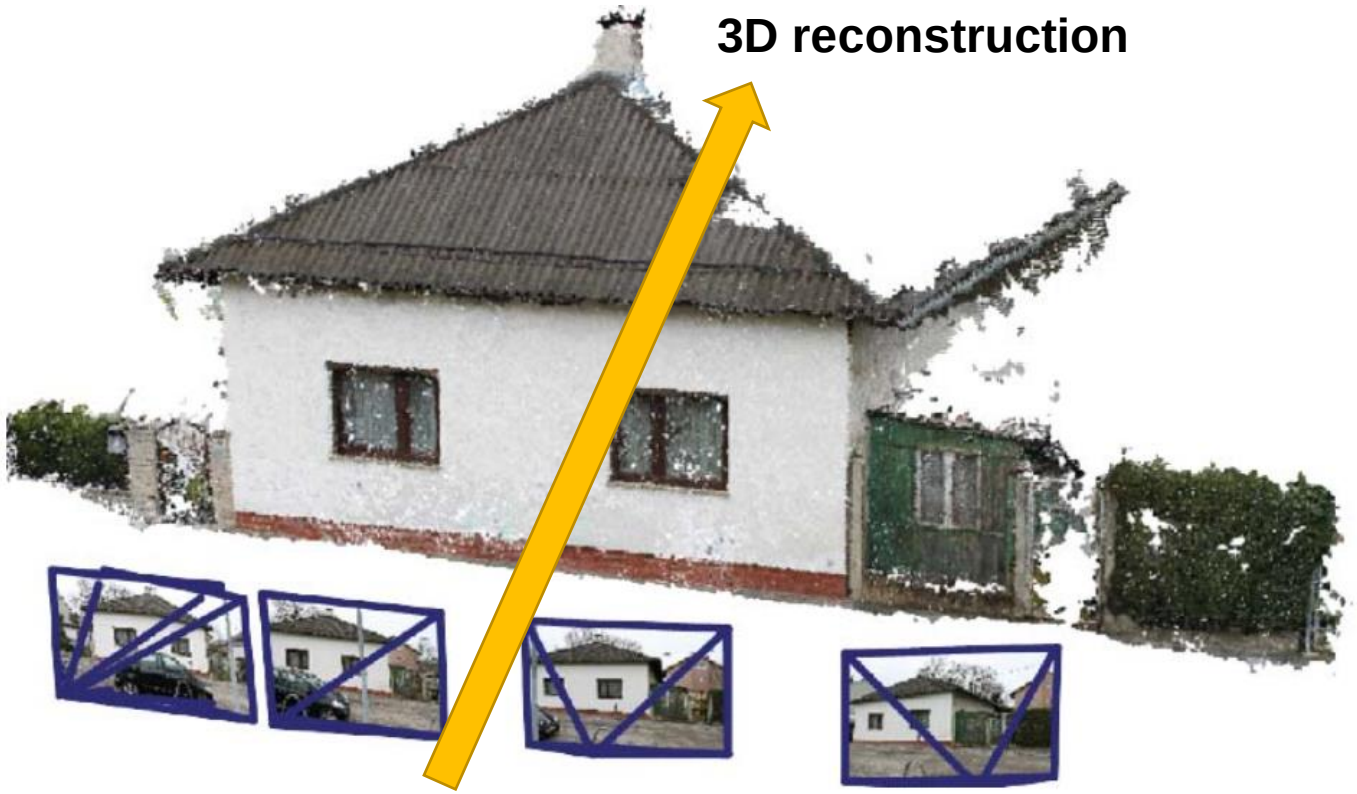
Learned statistics



Transposed correlation

Assignment on Image Collections

- Pull-push algorithm
- Image collection search
- Exemplar-based inpainting (*) -> higher difficulty

Next week

From collection of 2D images to 3D…