Introduction to Computer Vision: Image Transformations and Guided Image Filtering

Navasardyan Shant

September 26, 2019

Overview

1. Interpolation
   - Sampling
   - Interpolation
   - Coordinate Transformation

2. Guided Filtering
   - Applications
   - Algorithm
   - Tutorial on Guided Filter and Guided Upsampling

Overview

1. Interpolation
   - Sampling
   - Interpolation
   - Coordinate Transformation

2. Guided Filtering
   - Applications
   - Algorithm
   - Tutorial on Guided Filter and Guided Upsampling

Image Transformations

Geometric image transformations are commonly used technique in computer vision. Here You can see some of them:

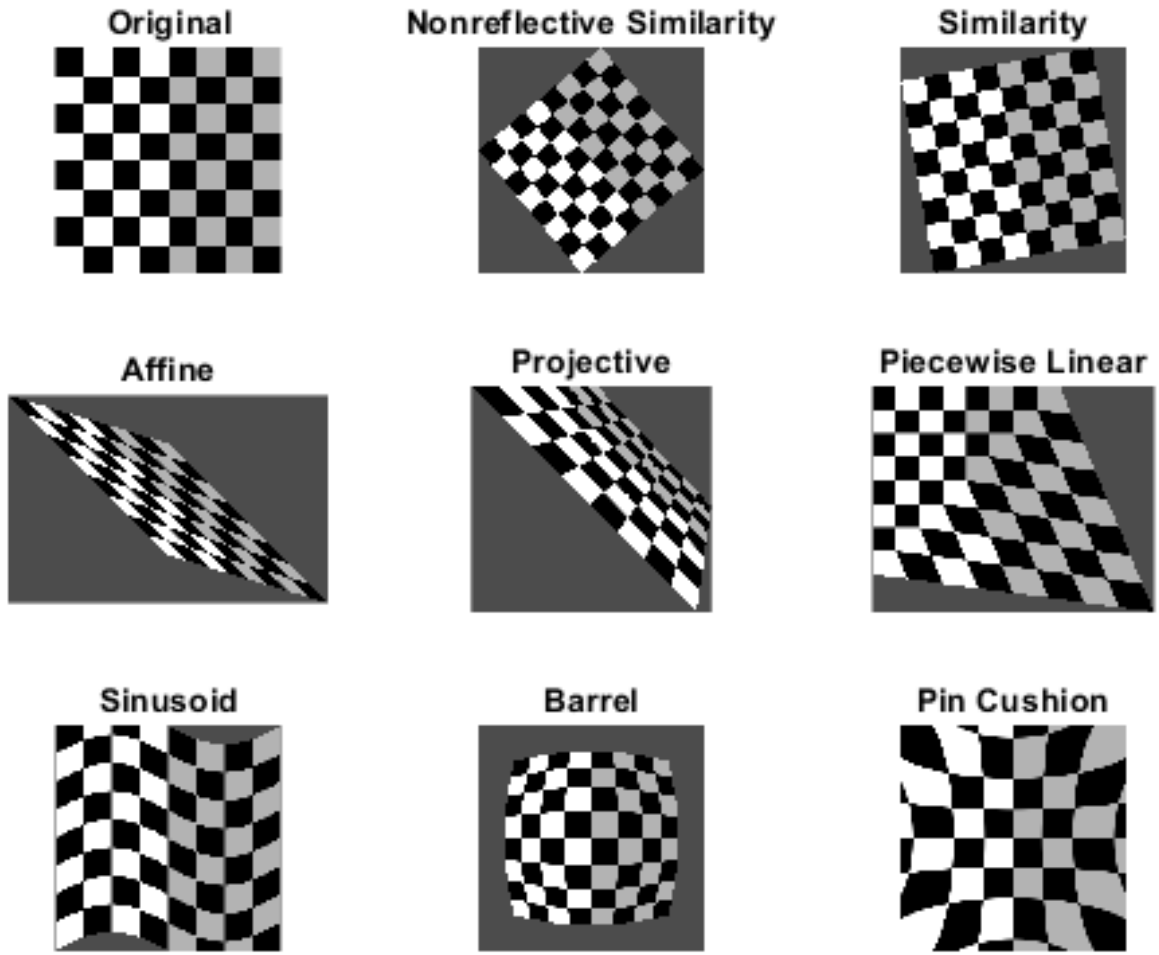


Image Transformations

What are image transformations? How we can implement them?

Image Transformations

What are image transformations? How we can implement them?

To answer these questions we need the following concepts:

- Interpolation
- Coordinate Transformation
- Sampling

Image Transformations

What are image transformations? How we can implement them?

To answer these questions we need the following concepts:

- Interpolation
- Coordinate Transformation
- Sampling

The Main Scheme

The following scheme shows a general approach for image transformations:

$$A \xrightarrow{\mathcal{I}} f \xrightarrow{\mathcal{T}} f' \xrightarrow{\mathcal{S}} A',$$

where $\mathcal{I}, \mathcal{T}$ and $\mathcal{S}$ are *interpolation*, *coordinate transformation* and *sampling* respectively.

Sampling

Sampling

Let we have a functional image $f : [0,1]^2 \to [0,1]$. We want to obtain a digital image $A \in \mathbb{R}^{H \times W}$ by using f. Then for any points $Q_{ij} \in [0,1]^2$ (for $i \in \{0, \ldots, H-1\}, j \in \{0, \ldots, W-1\}$) we can take $A_{ij} = f(Q_{ij})$. This process we call **sampling** from the function f.

Sampling

Sampling

Let we have a functional image $f : [0,1]^2 \to [0,1]$. We want to obtain a digital image $A \in \mathbb{R}^{H \times W}$ by using f. Then for any points $Q_{ij} \in [0,1]^2$ (for $i \in \{0, \dots, H-1\}, j \in \{0, \dots, W-1\}$) we can take $A_{ij} = f(Q_{ij})$. This process we call **sampling** from the function f.

Common Sampling

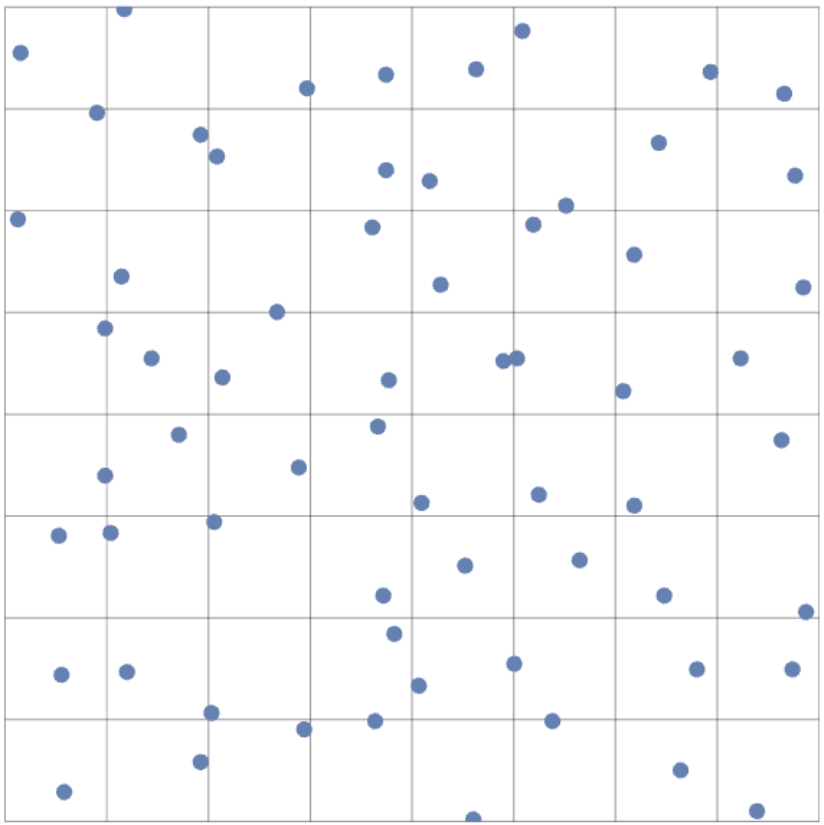
In most cases we will use the following sampling (uniform sampling):

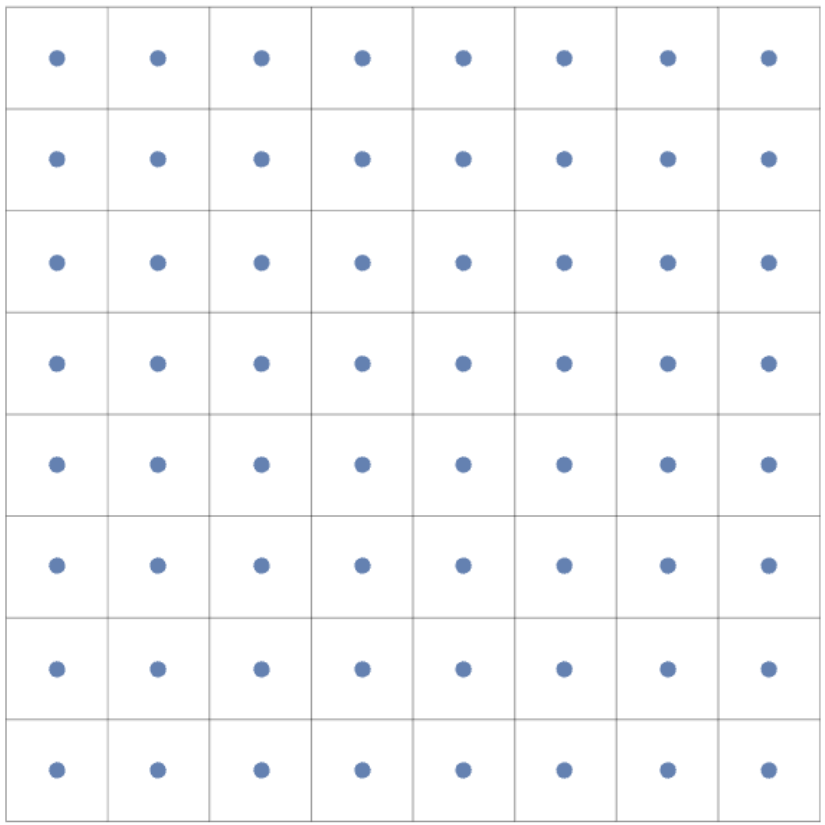
$$A_{ij} = f\left(\frac{i}{H} + \frac{1}{2H}, \frac{j}{W} + \frac{1}{2W}\right),$$

for $i \in \{0, 1, \dots, H-1\}$ and $j \in \{0, 1, \dots, W-1\}$.

Sampling

Below you can see two sampling methods: first is taking points Q_{ij} randomly each in ij-th cell of a $H \times W$-grid, second is taking points Q_{ij} as centers of the cells of the grid. Here $H = W = 8$.





Interpolation

In general, *interpolation* can be considered as the process of using known data to estimate values at unknown locations.

Interpolation

In general, *interpolation* can be considered as the process of using known data to estimate values at unknown locations.

Interpolation

Let $A \in \mathbb{R}^{H \times W}$ be an image, and $Q_{i,j} \in [0,1]^2$ are some distinct points for $i \in \{0, 1, \ldots, H-1\}, j \in \{0, 1, \ldots, W-1\}$. We want to find a function $f : [0,1]^2 \to [0,1]$ such that A is sampled from f on the points Q_{ij}, i.e.

$$A_{ij} = f(Q_{ij}) \quad \text{for all} \quad i, j.$$

Essentially this process is an estimation of values f from its values on specific points. This process we call **interpolation**.

Interpolation

In general, *interpolation* can be considered as the process of using known data to estimate values at unknown locations.

Interpolation

Let $A \in \mathbb{R}^{H \times W}$ be an image, and $Q_{i,j} \in [0,1]^2$ are some distinct points for $i \in \{0, 1, \ldots, H-1\}, j \in \{0, 1, \ldots, W-1\}$. We want to find a function $f : [0,1]^2 \to [0,1]$ such that A is sampled from f on the points Q_{ij}, i.e.

$$A_{ij} = f(Q_{ij}) \quad \text{for all} \quad i, j.$$

Essentially this process is an estimation of values f from its values on specific points. This process we call **interpolation**.

Note

In majority of cases we will need a function f such that A is uniformly sampled from f, so, if not mentioned otherwise, $Q_{i,j} = (\frac{i}{H} + \frac{1}{2H}, \frac{j}{W} + \frac{1}{2W})$.

Interpolation: Example

Nearest Neighbor Interpolation

Let $A \in \mathbb{R}^{H \times W}$ be an image. Let us consider the function $f : [0,1]^2 \to [0,1]$:

$$f(x,y) = \begin{cases} A_{[x \cdot H],[y \cdot W]}, & \text{if } x, y \in [0,1) \\ A_{[x \cdot H],W-1}, & \text{if } x \in [0,1), y = 1 \\ A_{H-1,[y \cdot W]} & \text{if } x = 1, y \in [0,1) \\ A_{H-1,W-1} & \textit{if} x = y = 1 \end{cases}.$$

This interpolation is called **Nearest Neighbor interpolation**.

Coordinate Transformation

Definition (Transformation)

Any bijective function $T : \mathbb{R}^2 \to \mathbb{R}^2$ is called a **transformation** of the plane $\mathbb{R}^2$.

Coordinate Transformation

Definition (Transformation)

Any bijective function $T : \mathbb{R}^2 \to \mathbb{R}^2$ is called a **transformation** of the plane $\mathbb{R}^2$.

Note

In this section we will talk about transformations $T : \mathbb{R}^2 \to \mathbb{R}^2$, for which the elements $T(x, y)$ might not be in $[0, 1]^2$ for some $x, y \in [0, 1]$. So here we enlarge the concept of an image to a function $\mathbb{R}^2 \to [0, 1]$. More precisely, let $f : [0, 1]^2 \to [0, 1]$ be an image, we identify this image to the function $\tilde{f} : \mathbb{R}^2 \to [0, 1]$, where

$$\tilde{f}(x, y) = \begin{cases} f(x, y), & \text{if } (x, y) \in [0, 1]^2 \\ 0, & \text{otherwise.} \end{cases}$$

As f is a restriction of $\tilde{f}$ on $[0, 1]^2$, we will omit $\sim$, where it is clear.

Image Transformations for Functions

Note

Since we have defined image also as a function on $\mathbb{R}^2$, we call any plane transformation **coordinate transformation**.

Image Transformations for Functions

Note

Since we have defined image also as a function on $\mathbb{R}^2$, we call any plane transformation **coordinate transformation**.

Note

Let $f : \mathbb{R}^2 \to [0, 1]$ be an image, $T : \mathbb{R}^2 \to \mathcal{R}^2$ be a coordinate transformation. One can define the function $f' = T^{-1} \circ f$:

$$f'(x', y') = f(T^{-1}(x', y')) \quad \text{for } x', y' \in \mathbb{R}.$$

The procedure $\mathcal{T}$ (accociated with the coordinate transformation T) of obtaining the function f' from the function f we call **image transformation**. For obtaining digital image transformations, we need also sampling and interpolation.

Coordinate Transformation: Affine Transformations

In this course we concentrate on *affine transformations*.

Definition (affine transformation)

The transformation $\mathcal{A} : \mathbb{R}^2 \to \mathbb{R}^2$ which corresponds the element $(x, y) \in \mathbb{R}^2$ to the element $\mathcal{A}(x', y') \in \mathbb{R}^2$ by the following way:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = A \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix},$$

is called **affine transformation** (of the plane $\mathbb{R}^2$). The matrix A is called the **affine transformation matrix**.

Coordinate Transformation: Affine Transformations

In this course we concentrate on *affine transformations*.

Definition (affine transformation)

The transformation $\mathcal{A} : \mathbb{R}^2 \to \mathbb{R}^2$ which corresponds the element $(x, y) \in \mathbb{R}^2$ to the element $\mathcal{A}(x', y') \in \mathbb{R}^2$ by the following way:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = A \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix},$$

is called **affine transformation** (of the plane $\mathbb{R}^2$). The matrix A is called the **affine transformation matrix**.

We will consider algorithms of a special case of affine transformation - rotation transformation.

Special Case: Rotation

Definition (rotation)

Let $\theta \in [0, 2\pi]$. Then the affine transformation with the following affine transformation matrix is called **rotation**:

$$A_\theta = \begin{pmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Hence, for rotating the image $f : \mathbb{R}^2 \to [0, 1]$ with an angle θ, we need to do following: compute the matrix A_θ^{-1}, then define a function $f' : \mathbb{R}^2 \to [0, 1]$, $\quad f'(x', y') = f(x, y)$, where $(x, y, 1)^T = A_\theta^{-1}(x', y', 1)^T$.

Special Case: Rotation

Recall the scheme of image transformations:

$$A \xrightarrow{\mathcal{I}} f \xrightarrow{\mathcal{T}} f' \xrightarrow{\mathcal{S}} A'.$$

In case of rotation transformation we take

- some interpolation $\mathcal{I}$,
- the image rotation transformation $\mathcal{A}_\theta(f)(x', y') = f(x, y)$, where $(x, y, 1)^T = A_\theta^{-1}(x', y', 1)^T$,
- the uniform sampling $\mathcal{S}(g)_{i,j} = g(\frac{i}{H} + \frac{1}{2H}, \frac{j}{W} + \frac{1}{2W})$.

Special Case: Rotation

Recall the scheme of image transformations:

$$A \xrightarrow{\mathcal{I}} f \xrightarrow{\mathcal{T}} f' \xrightarrow{\mathcal{S}} A'.$$

In case of rotation transformation we take

- some interpolation $\mathcal{I}$,
- the image rotation transformation $\mathcal{A}_\theta(f)(x', y') = f(x, y)$, where $(x, y, 1)^T = A_\theta^{-1}(x', y', 1)^T$,
- the uniform sampling $\mathcal{S}(g)_{i,j} = g(\frac{i}{H} + \frac{1}{2H}, \frac{j}{W} + \frac{1}{2W})$.

Recall that the nearest neighbor interpolation of f from the array $A \in \mathbb{R}^{H \times W}$ (for uniform sampling) is defined as

$$f(x, y) = A_{[xH],[yW]}, \quad x, y \in [0, 1).$$

Rotation: Exercise

Exercise

Implement the image rotation transformation with nearest neighbor interpolation.

Rotation: Exercise

Exercise

Implement the image rotation transformation with nearest neighbor interpolation.

Nearest Neighbour Algorithm for Rotation

Let $A = (a_{ij})^{H \times W}$ be an image, $\theta \in [0, 2\pi]$ be an angle. Let's obtain the image A' by rotating the image A by the angle θ. At first we define the function $f'(x', y') = f_A(x, y)$, where $(x, y, 1)^T = A_\theta^{-1}(x', y', 1)^T$, so we get

$$f'(x', y') = f_A(x, y) = a_{[x \cdot H], [y \cdot W]}, \quad \text{if x, y} \in [0, 1),$$

where $x = x'cos\theta + y'sin\theta, y = y'cos\theta - x'sin\theta$. It remains to take $A' = (f'(\frac{i'}{H} + \frac{1}{2H}, \frac{j'}{W} + \frac{1}{2W}))^{H \times W}$.

Rotation: Formula

Formula

$$A'_{ij} = \mathcal{S}(\mathcal{A}_\theta(\mathcal{NN}(A)))_{ij} = \mathcal{A}_\theta(\mathcal{NN}(A))\left(\frac{i}{H} + \frac{1}{2H}, \frac{j}{W} + \frac{1}{2W}\right) =$$

$$\mathcal{NN}(A)\left(\tilde{A}_\theta^{-1}\left(\frac{i}{H} + \frac{1}{2H}, \frac{j}{W} + \frac{1}{2W}, 1\right)^T\right)$$

$$= A_{[(H,W)*\tilde{A}_\theta^{-1}\left(\frac{i}{H}+\frac{1}{2H}, \frac{j}{W}+\frac{1}{2W}\right)^T]},$$

if $A_{[(H,W)*\tilde{A}_\theta^{-1}\left(\frac{i}{H}+\frac{1}{2H}, \frac{j}{W}+\frac{1}{2W}\right)^T]}$ is defined and 0 otherwise. (Here $\tilde{A}_\theta = A_\theta[:-1, :-1]$, $*$ is the element-wise multiplication and $[(u, v)] = ([u], [v])$.)

Rotation: Solution

```python
import numpy as np
def rotate(img, alpha, b = [0,0]):
    A_alpha = np.array([[np.cos(alpha), -np.sin(alpha)],
                        [np.sin(alpha), np.cos(alpha)]
                        ])
    A_alpha_inv = np.linalg.inv(A_alpha)

    H, W = img.shape
    A_p = np.zeros_like(img)
    for i in range(H):
        for j in range(W):
            x, y = np.dot(A_alpha_inv, [(2*i+1)/(2*H) - b[0], (2*j+1)/(2*W) - b[1]])
            x, y = x + b[0], y + b[1]
            if 0<=x<1 and 0<=y<1:
                A_p[i,j] = img[int(x*H), int(y*W)]
    return A_p
```

Lagrange Polynomial for Interpolation

Now we will consider some other interpolation methods.

Lagrange Polynomial for Interpolation

Now we will consider some other interpolation methods.

Recall

Let

$$-\infty < a = x_0 < x_1 < \ldots < x_n = b < +\infty,$$

and we have that a function $f : [a, b] \to \mathbb{R}$ is given on the points $x_0, \ldots, x_n$. We need to estimate vales of f for any $x \in [a, b]$. Then the Lagrange polynomial gives us such estimation $p(x) \approx f(x)$:

$$p(x) = \sum_{i=0}^{n} f(x_i) \frac{(x - x_0) \ldots (x - x_{i-1})(x - x_{i+1}) \ldots (x - x_n)}{(x_i - x_0) \ldots (x_i - x_{i-1})(x_i - x_{i+1}) \ldots (x_i - x_n)}.$$

Note that $deg(p) = n$. When $n = 1$, we call this interpolation **linear**.

Bilinear Interpolation

So, the linear interpolation of the function $f : [a, b] \to \mathbb{R}$ for points $a, b \in \mathbb{R}$ is the function

$$p(x) = f(a)\frac{x-b}{a-b} + f(b)\frac{x-a}{b-a} = \frac{f(b)-f(a)}{b-a}x + \frac{bf(a)-af(b)}{b-a}$$

Bilinear Interpolation

So, the linear interpolation of the function $f : [a, b] \to \mathbb{R}$ for points $a, b \in \mathbb{R}$ is the function

$$p(x) = f(a)\frac{x-b}{a-b} + f(b)\frac{x-a}{b-a} = \frac{f(b)-f(a)}{b-a}x + \frac{bf(a)-af(b)}{b-a}$$

Bilinear Interpolation

Let $x_1 < x_2$, and $y_1 < y_2$, consider the points $Q_{11} = (x_1, y_1), Q_{12} = (x_1, y_2), Q_{21} = (x_2, y_1), Q_{22} = (x_2, y_2)$. Let we have values of a function $f : [x_1, x_2] \times [y_1, y_2] \to \mathbb{R}$ on the points Q_{ij}. We need to estimate values $f(x, y)$ for $(x, y) \in [x_1, x_2] \times [y_1, y_2]$.

Bilinear Interpolation

First we can fix y_1, y_2 and do linear interpolation for functions $f(\cdot, y_1)$ and $f(\cdot, y_2)$:

$$p(x, y_1) = f(Q_{11})\frac{x - x_2}{x_1 - x_2} + f(Q_{21})\frac{x - x_1}{x_2 - x_1},$$

$$p(x, y_2) = f(Q_{12})\frac{x - x_2}{x_1 - x_2} + f(Q_{22})\frac{x - x_1}{x_2 - x_1}.$$

Bilinear Interpolation

First we can fix y_1, y_2 and do linear interpolation for functions $f(\cdot, y_1)$ and $f(\cdot, y_2)$:

$$p(x, y_1) = f(Q_{11})\frac{x - x_2}{x_1 - x_2} + f(Q_{21})\frac{x - x_1}{x_2 - x_1},$$

$$p(x, y_2) = f(Q_{12})\frac{x - x_2}{x_1 - x_2} + f(Q_{22})\frac{x - x_1}{x_2 - x_1}.$$

Then for fixed $x \in [x_1, x_2]$ we can do linear interpolation for the function $p(x, \cdot)$:

$$q(x, y) = p(x, y_1)\frac{y - y_2}{y_1 - y_2} + p(x, y_2)\frac{y - y_1}{y_2 - y_1},$$

and finally, we do estimation $f(x, y) \approx q(x, y)$.
This interpolation of the function $f(x, y)$ is called **bilinear interpolation**.

Note

If an image $A \in \mathbb{R}^{H \times W}$ is given, we can get the biliner interpolated function-image f from $f(Q_{ij}) = A_{ij}$ for uniform sampling Q_{ij}.

Bilinear Interpolation

One can find out that

$$f(x,y) \approx q(x,y) =$$

$$\frac{1}{(x_2-x_1)(y_2-y_1)}(x_2-x \quad x-x_1)\begin{pmatrix} f(Q_{11}) & f(Q_{12}) \\ f(Q_{21}) & f(Q_{22}) \end{pmatrix}\begin{pmatrix} y_2-y \\ y-y_1 \end{pmatrix}$$

Bilinear Interpolation

One can find out that

$$f(x,y) \approx q(x,y) =$$

$$\frac{1}{(x_2-x_1)(y_2-y_1)}\begin{pmatrix} x_2-x & x-x_1 \end{pmatrix}\begin{pmatrix} f(Q_{11}) & f(Q_{12}) \\ f(Q_{21}) & f(Q_{22}) \end{pmatrix}\begin{pmatrix} y_2-y \\ y-y_1 \end{pmatrix}$$

Exercise

Implement rotation with bilinear interpolation.

Bilinear Interpolation

One can find out that

$$f(x,y) \approx q(x,y) =$$

$$\frac{1}{(x_2-x_1)(y_2-y_1)}\begin{pmatrix} x_2-x & x-x_1 \end{pmatrix}\begin{pmatrix} f(Q_{11}) & f(Q_{12}) \\ f(Q_{21}) & f(Q_{22}) \end{pmatrix}\begin{pmatrix} y_2-y \\ y-y_1 \end{pmatrix}$$

Exercise

Implement rotation with bilinear interpolation.

Note

We can obtain the same bilinear interpolation algorithm by the following way: let us approximate the function $f(x,y) \approx q(x,y) = a_0 + a_1x + a_2y + a_3xy$. Then we can simply find a_i by solving the system of linear equations $q(Q_{ij}) = f(Q_{ij})$.
Note that here we can either take q(x,y) = 0, if $(x,y) \notin [x_0, x_{H-1}] \times [y_0, y_{W-1}]$ or just take Q_{ij} as the closest 4 points of (x,y).

Bicubic Interpolation

As in case of bilinear interpolation, here we consider the approximation of the function $f(x, y) \approx q(x, y)$ as

$$q(x, y) = \sum_{i=0}^{3} \sum_{j=0}^{3} q_{ij} x^i y^j.$$

Bicubic Interpolation

As in case of bilinear interpolation, here we consider the approximation of the function $f(x, y) \approx q(x, y)$ as

$$q(x, y) = \sum_{i=0}^{3} \sum_{j=0}^{3} q_{ij} x^i y^j.$$

We find the coefficients q_{ij} from 16 linear equations $q(Q_{ij}) = f(Q_{ij})$, where Q_{ij} are the 16 closest points of (x, y).

Bicubic Interpolation

As in case of bilinear interpolation, here we consider the approximation of the function $f(x, y) \approx q(x, y)$ as

$$q(x, y) = \sum_{i=0}^{3} \sum_{j=0}^{3} q_{ij} x^i y^j.$$

We find the coefficients q_{ij} from 16 linear equations $q(Q_{ij}) = f(Q_{ij})$, where Q_{ij} are the 16 closest points of (x, y).

Definition (Bicubic Interpolation)

This interpolation of the function $f(x, y)$ is called **bicubic interpolation**.

The Upsampling Problem

Let $A \in \mathbb{R}^{H \times W}$, we want to resize the image to the size for example $2H \times 2W$.

Figure: Resized 4 times by 2 different resize methods

The Upsampling Problem

We can approach to this problem by the same scheme as for image transformations:

The Upsampling Problem

We can approach to this problem by the same scheme as for image transformations:

Scheme for upsampling

Let $A \in \mathbb{R}^{H \times W}$ be an image, we want to obtain its resized version of size $H' \times W'$, let us denote it $A' \in \mathbb{R}^{H' \times W'}$. We can follow the scheme

$$A \xrightarrow{\mathcal{I}} f \xrightarrow{\mathcal{T}} f' \xrightarrow{\mathcal{S}} A',$$

where the image transformation operator $\mathcal{T}$ is identical, i.e. $f' = \mathcal{T}(f) = f$. Then we can simply choose an interpolation and uniform sampling of size $H' \times W'$.

The Upsampling Problem

We can approach to this problem by the same scheme as for image transformations:

Scheme for upsampling

Let $A \in \mathbb{R}^{H \times W}$ be an image, we want to obtain its resized version of size $H' \times W'$, let us denote it $A' \in \mathbb{R}^{H' \times W'}$. We can follow the scheme

$$A \xrightarrow{\mathcal{I}} f \xrightarrow{\mathcal{T}} f' \xrightarrow{\mathcal{S}} A',$$

where the image transformation operator $\mathcal{T}$ is identical, i.e. $f' = \mathcal{T}(f) = f$. Then we can simply choose an interpolation and uniform sampling of size $H' \times W'$.

Exercise

Implement image resize with nearest neighbor/bilinear interpolation.

Nearest Neighbor Upsampling: Implementation

```python
import numpy as np
def resize_NN(img, shape):
    H, W = img.shape
    H_n, W_n = shape
    A_p = np.zeros(shape)
    for i in range(H_n):
        for j in range(W_n):
            x, y = (2*i+1)/(2*H_n), (2*j+1)/(2*W_n)
            A_p[i,j] = img[int(x*H), int(y*W)]
    return A_p
```

Overview

1. Interpolation
   - Sampling
   - Interpolation
   - Coordinate Transformation

2. Guided Filtering
   - Applications
   - Algorithm
   - Tutorial on Guided Filter and Guided Upsampling

Guided Image Filtering

Guided Image Filtering

Guided Image Filtering is an algorithm to obtain some image q from an *input image* p and a *guidance image* I.

Guided Image Filtering

Guided Image Filtering

Guided Image Filtering is an algorithm to obtain some image q from an *input image* p and a *guidance image* I.

More strictly, guided image filtering is an algorithm to determine the coefficients $W_{ij} = W_{ij}(I)$ in order to get plausible result q, ith pixel of which is defined by

$$q_i = \sum_j W_{ij}(I) p_j.$$

Guided Image Filtering

Guided Image Filtering

Guided Image Filtering is an algorithm to obtain some image q from an *input image* p and a *guidance image* I.

More strictly, guided image filtering is an algorithm to determine the coefficients $W_{ij} = W_{ij}(I)$ in order to get plausible result q, ith pixel of which is defined by

$$q_i = \sum_j W_{ij}(I) p_j.$$

First we discuss some applications of guided image filtering.

Guided Image Filtering

Guided Image Filtering

Guided Image Filtering is an algorithm to obtain some image q from an *input image* p and a *guidance image* I.

More strictly, guided image filtering is an algorithm to determine the coefficients $W_{ij} = W_{ij}(I)$ in order to get plausible result q, ith pixel of which is defined by

$$q_i = \sum_j W_{ij}(I) p_j.$$

First we discuss some applications of guided image filtering.

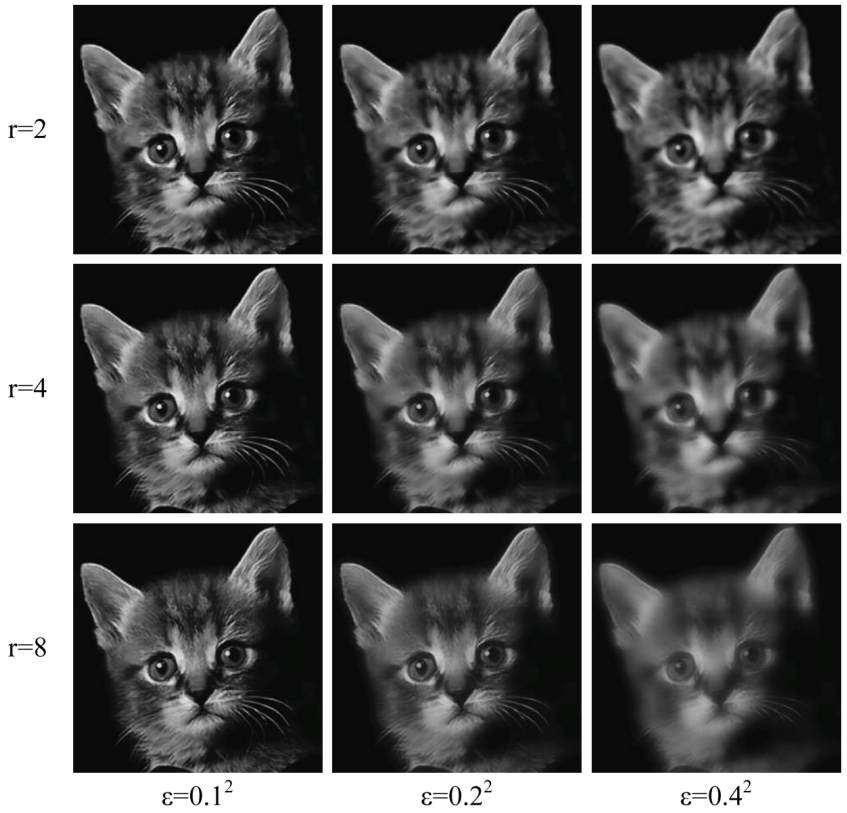
paper

The link of the original paper of guided image filtering:
http://kaiminghe.com/publications/pami12guidedfilter.pdf

Edge-Preserving Smoothing

Edge-preserving smooth image can be obtained with guided filter by taking the input image also as guidance image. Here are some results with different parameters of guided filter:

input



Detail Enhancement

Detail enhancement is the task of making the details in the image more expressive.

Original Guided Filter

Image Matting

The task of *image matting* is to predict the opacity of each pixel for foreground object. If the foreground object is roughly detected, then guided filter can be applied to get detailed foreground-detection.

filtering input

guide I

Guided Filter

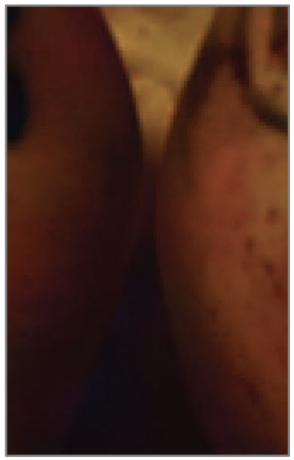
Flash/No-Flash Denoising

Let we have two photos one with flash and the other without flash but with noise. We want to obtain an image without noise and without flash.

Filter Input p

Guidance I

Guided Filter



Guided Image Filtering: The Algorithm

As we know, guided image filtering obtains an image q from an input image p and a guidance image I.

Guided Image Filtering: The Algorithm

As we know, guided image filtering obtains an image q from an input image p and a guidance image I.

The idea of guided filter is the local linear dependency between the guidance image I and the filter output q. More precisely, we assume that for pixel k in q for some window ω_k

$$q_i = a_k I_i + b_k, \quad \text{for all} \quad i \in \omega_k.$$

Guided Image Filtering: The Algorithm

As we know, guided image filtering obtains an image q from an input image p and a guidance image I.

The idea of guided filter is the local linear dependency between the guidance image I and the filter output q. More precisely, we assume that for pixel k in q for some window ω_k

$$q_i = a_k I_i + b_k, \quad \text{for all} \quad i \in \omega_k.$$

Note

Let us notice that if we make such an assumption for every pixel k, and suppose that a_k, b_k does not depend on i, we will have different values for the same q_i. So as you can see later we do not make the assumption of independence of a_k, b_k from i, and this "local linearity" is just for idea of guided filtering.

Guided Image Filtering: The Algorithm

So we assume that $q_i = a_k I_i + b_k$ for each $i \in \omega_k$. Let's determine the coefficients a_k, b_k.

Guided Image Filtering: The Algorithm

So we assume that $q_i = a_k I_i + b_k$ for each $i \in \omega_k$. Let's determine the coefficients a_k, b_k.

Note

As one of the problems we want to solve with guided filter is the denoising problem (as well as edge-preserving smoothing), we model the input image p as a composition of the wanted image q and some noise n: $q_i = p_i - n_i$.

Guided Image Filtering: The Algorithm

So we assume that $q_i = a_k I_i + b_k$ for each $i \in \omega_k$. Let's determine the coefficients a_k, b_k.

Note

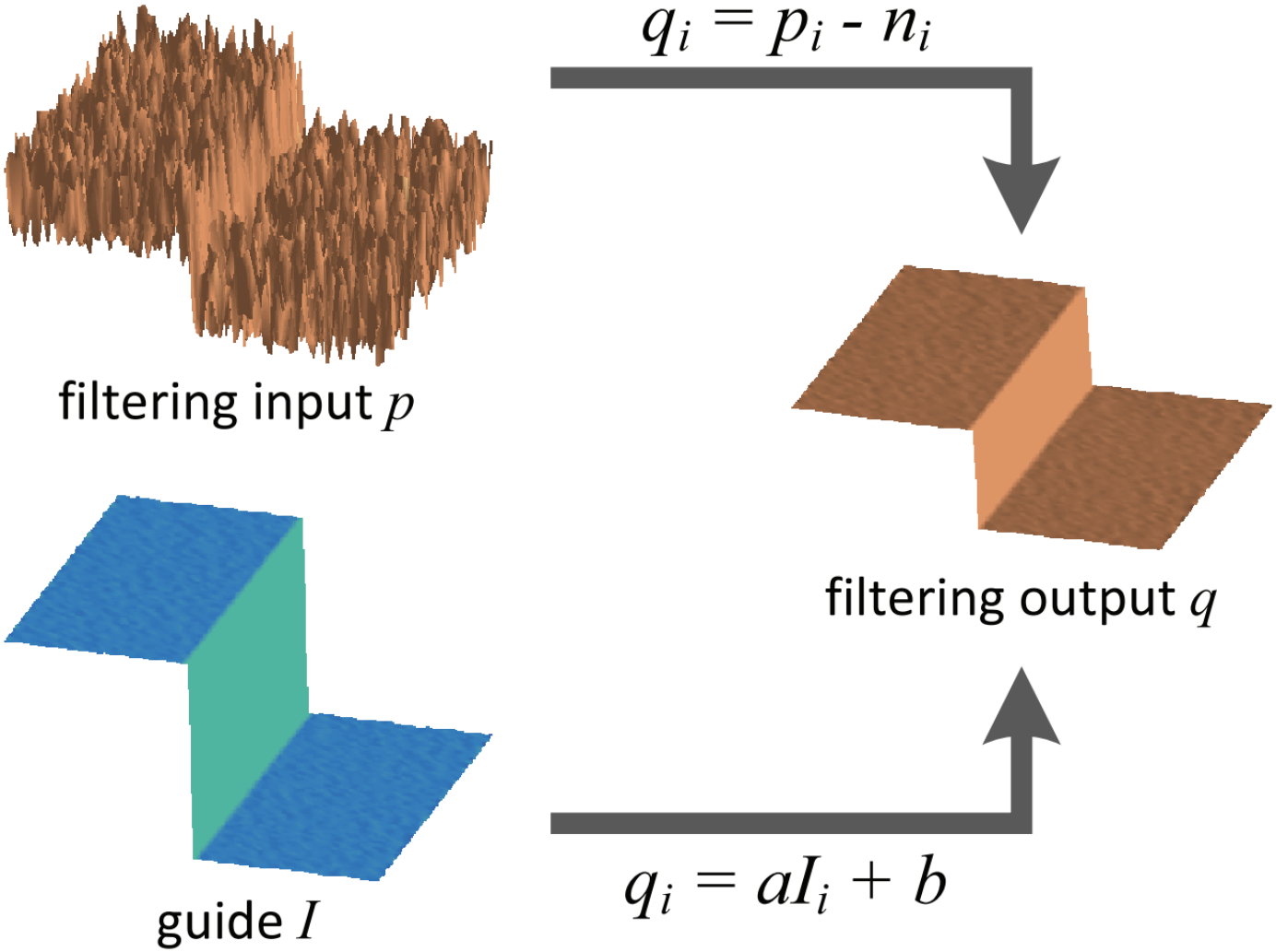
As one of the problems we want to solve with guided filter is the denoising problem (as well as edge-preserving smoothing), we model the input image p as a composition of the wanted image q and some noise n: $q_i = p_i - n_i$.

Note

Hence, on the one hand we want the noise to become small, so we minimize the component $(a_k I_i + b_k - p_i)^2$ (in order to preserve structures of p), on the other hand we want to give not very large coefficient to the I_i in the formula $q_i = a_k I_i + b_k$ (Since for example in the case $I = p$, we want to preserve structure but get rid of noise), so we also minimize the component εa_k^2 for some parameter ε.

Guided Image Filtering: The Algorithm

The following image is illustrating the guided filtering process for denoising (edge-preserving smoothing):



Guided Image Filtering: The Algorithm

So we minimize the following cost function w.r.t. a_k, b_k:

Guided Image Filtering: The Algorithm

So we minimize the following cost function w.r.t. a_k, b_k:

$$E(a_k, b_k) = \sum_{i \in \omega_k} ((a_k I_i + b_k - p_i)^2 + \varepsilon a_k^2).$$

Guided Image Filtering: The Algorithm

So we minimize the following cost function w.r.t. a_k, b_k:

$$E(a_k, b_k) = \sum_{i \in \omega_k} ((a_k I_i + b_k - p_i)^2 + \varepsilon a_k^2).$$

This function has global minimum, which can be given explicitly with the formula

$$a_k = \frac{\frac{1}{|\omega_k|} \sum_{j \in \omega_k} I_j p_j - \mu_k \overline{p}_k}{\sigma_k^2 + \varepsilon}, \qquad b_k = \overline{p}_k - a_k \mu_k,$$

where μ_k and σ_k^2 are the mean and variance of I in ω_k, $|\omega_k|$ is the number of pixels in ω_k, and $\overline{p}_k$ is the mean of p in ω_k.
Since with the formula $q_i = a_k I_i + b_k$ we will have many possible values of q_i, we simply take the average of this values: $q_i = \frac{1}{|\omega_k|} \sum_{k: i \in \omega_k} (a_k I_i + b_k)$.

Guided Image Filtering: The Algorithm

Guided Filter: Definition

So we get the output of guided filter:

$$q_i = \bar{a}_i I_i + \bar{b}_i,$$

where

$$\bar{a}_i = \frac{1}{|\omega_i|} \sum_{k \in \omega_i} a_k, \quad \bar{b}_i = \frac{1}{|\omega_i|} \sum_{k \in \omega_i} b_k$$

Guided Image Filtering: The Algorithm

Guided Filter: Definition

So we get the output of guided filter:

$$q_i = \bar{a}_i I_i + \bar{b}_i,$$

where

$$\bar{a}_i = \frac{1}{|\omega_i|} \sum_{k \in \omega_i} a_k, \quad \bar{b}_i = \frac{1}{|\omega_i|} \sum_{k \in \omega_i} b_k$$

Let's proceed with tutorial of guided filtering and guided upsampling.

Introduction to Computer Vision: Image Transformations and Guided Image Filtering

Navasardyan Shant

September 26, 2019