Image Processing
실습 5주차

안 준 혁

Department of Computer Science and Engineering

Chungnam National University, Korea

실습 소개

- **과목 홈페이지**
  - 충남대학교 사이버 캠퍼스 (http://e-learn.cnu.ac.kr)

- **TA 연락처**
  - 공대 5호관 531호 컴퓨터비전 연구실
  - 과제 질문은 [IP]를 제목에 붙여 메일로 주세요.
  - 00반
    - 안준혁
    - ajh99345@gmail.com
  - 01반
    - 신동헌
    - doghon85@naver.com

목차

- **3주차 과제 리뷰**
- **실습**
  - Image downsampling
    - Naïve downsampling
    - Gaussian downsampling
- **과제**
  - Image resizing

3주차 과제 리뷰

- **Histogram equalization**
  1. Histogram을 구한다.
  2. Histogram을 총 픽셀로 나눈다.
  3. 누적 분포(CDF)를 구한다. (my_PDF2CDF)
  4. 누적 시킨 값에 gray level의 최댓값(255)을 곱한다.
  5. 반올림 또는 버림을 하여 정수 값으로 변환한다.
  6. 구해진 정수 값을 사용하여 histogram equalization 결과를 반환한다.

input gray level k	0	1	2	3	4	5	6	7
normalized input r_k	0	1/7	2/7	3/7	4/7	5/7	6/7	1
histogram n_k	1	3	2	7	8	3	0	1
normalized histogram n_k/N	1/25	3/25	2/25	7/25	8/25	3/25	0	1/25
normalized output s_k	1/25	4/25	6/25	13/25	21/25	24/25	24/25	1
denormalized output $o_k = s_k \times 7$	7/25	28/25	42/25	91/25	147/25	168/25	168/25	7
output gray level floor(o_k)	0	1	1	3	5	6	6	7
m	0	1	2	3	4	5	6	7
output histogram n_m	1	5	0	7	0	8	3	1

Image downsampling

- **Naïve image downsampling**
  - 방법: 특정 비율 만큼의 행과 열을 삭제

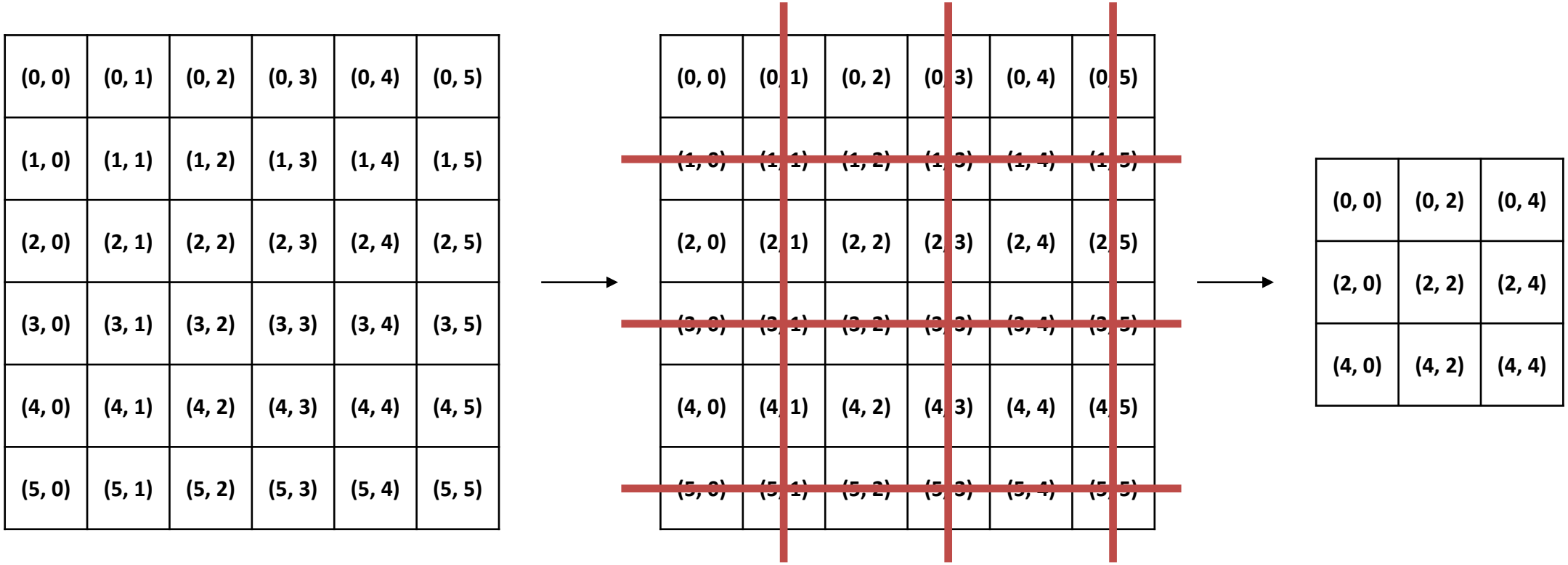
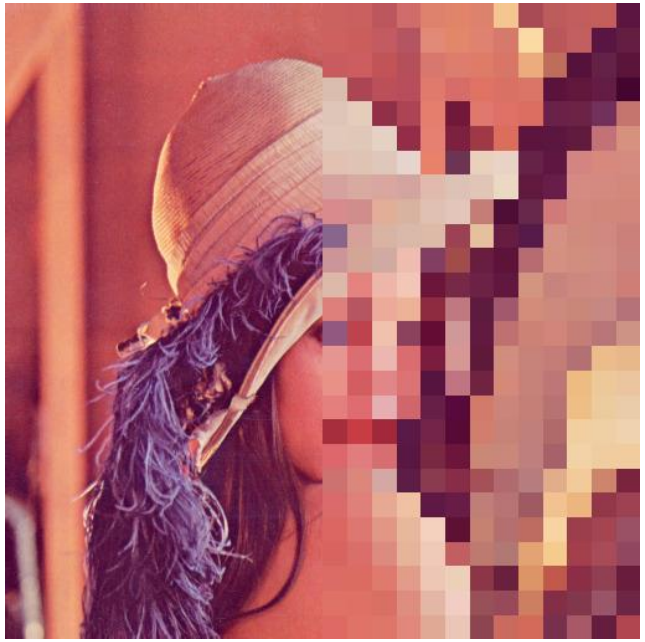
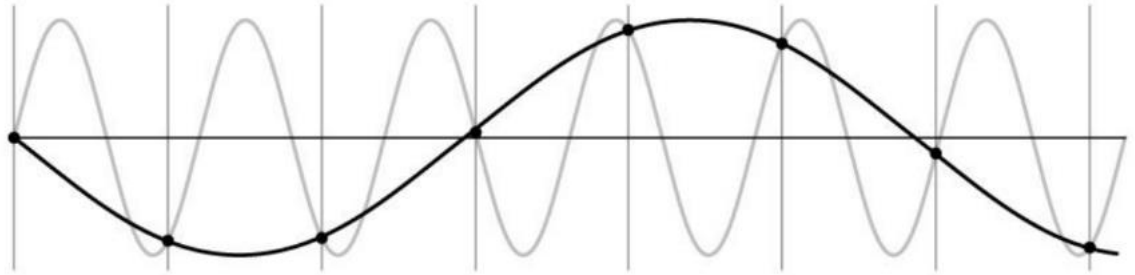


Image downsampling

- **Naïve image downsampling**
  - 문제점: Pixelated(Aliasing)
  - 해결방안: Smoothing 기법을 통한 Filtering



Pixelated



Aliasing

실습 1 my_naïve_downsampling

- **main():**
  1. old_img: OpenCV를 활용하여 Lena.png를 grayscale로 불러오기
  2. new_img: my_naïve_downsampling을 이용하여 old_img를 downsampling
  3. OpenCV를 이용하여 new_img 저장
- **my_naïve_downsampling(old_img, ratio_y, ratio_x):**
  - old_img: 불러온 이미지
  - ratio_y: 남길 행 비율
  - ratio_x: 남길 열 비율
  - return new_img: downsampling된 이미지

Image downsampling

- **Smoothing 기법 이후 image downsampling**
  - Smoothing 기법: average filtering, gaussian filtering, …
  - Pixelated 문제점은 해결했지만 smoothing 기법으로 인한 정보손실은 어쩔 수 없음

Naïve downsampling

Gaussian downsampling

실습 2 downsampling_after_smoothing.py

- **main():**
  1. old_img: OpenCV를 활용하여 Lena.png를 grayscale로 불러오기
  2. filter: average filter 혹은 gaussian filter 만들기
  3. filtered_img: cv2.filter2D(src, ddepth, kernel, border_Type)사용
     - src: old_img
     - ddepth: 출력 영상의 데이터 타입을 지정 -1이면 입력과 동일
     - kernel: filter
     - border_Type: cv2.BORDER_CONSTANT로 zero_padding
  4. new_img: my_naïve_downsampling을 이용하여 filtered_img를 downsampling
  5. OpenCV를 이용하여 new_img 저장

Image resizing

- **Resize 과제**
  - Upsampling

Image resizing

- **Resize 과제**
  - Downsampling

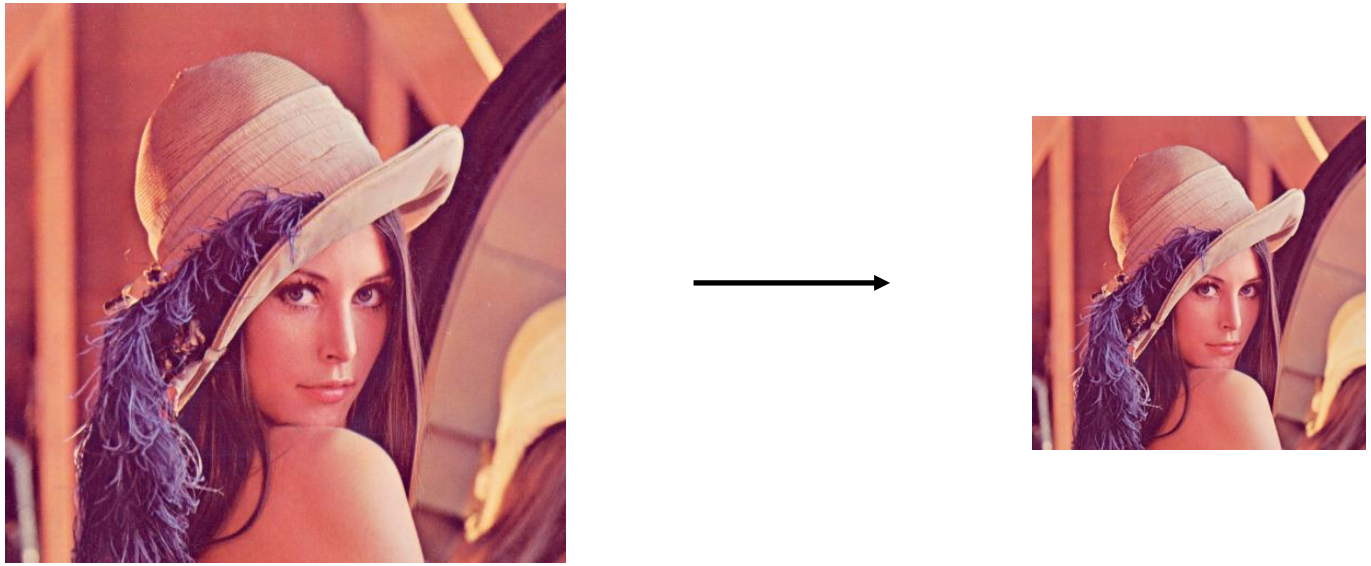
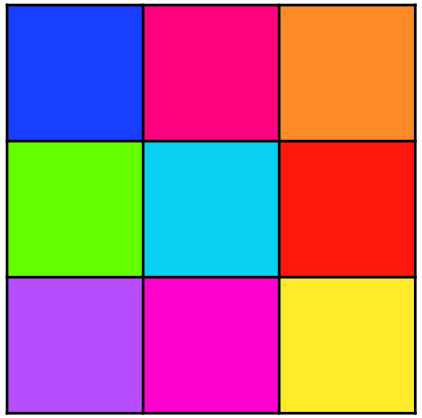
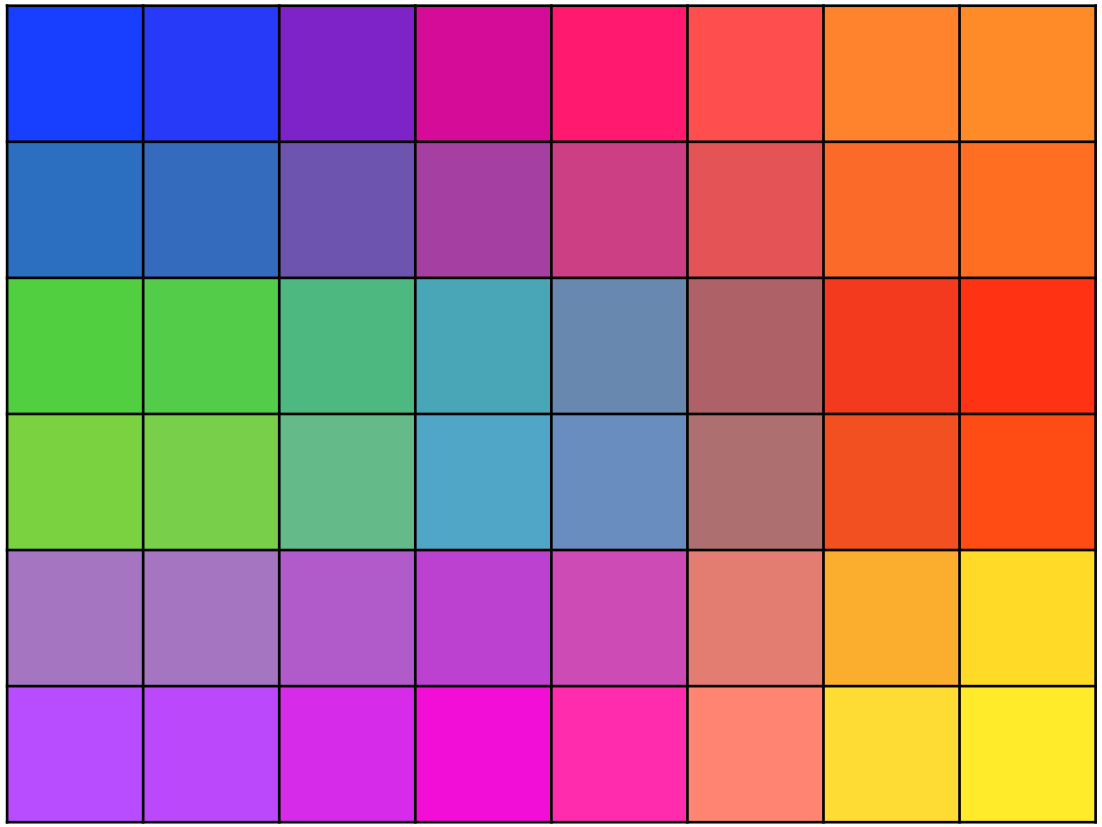


Image resizing

- **Resize 3x3 → 6x8**
  - Image upsampling



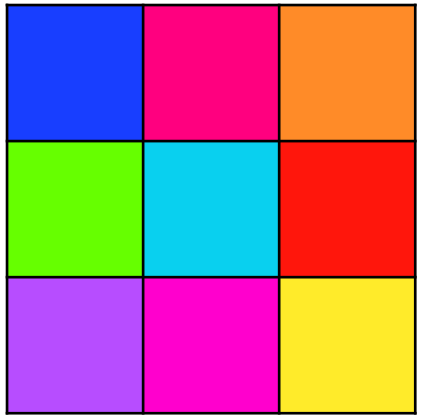
3x3 image
(Old)



6x8 image
(New)

Image resizing

- **Resize 3x3 → 6x8**
  1. Create new image
     - (6, 8)의 빈 배열 만들기 (0으로 채우기)



3x3 image
(Old)

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

6x8 image
(New)

Image resizing

- **Resize 3x3 → 6x8**
  2. Match up coordinates

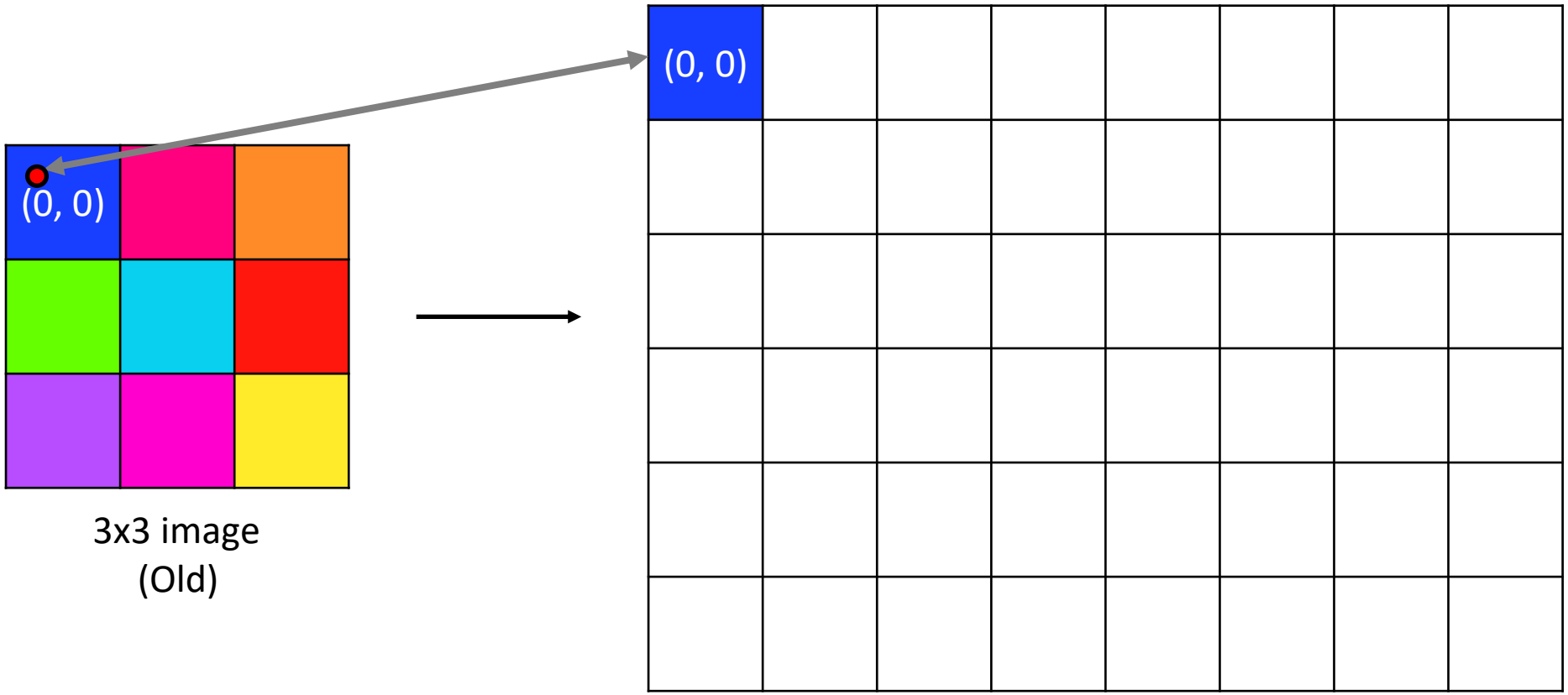


Image resizing

- **Resize 3x3 → 6x8**
  2. Match up coordinates

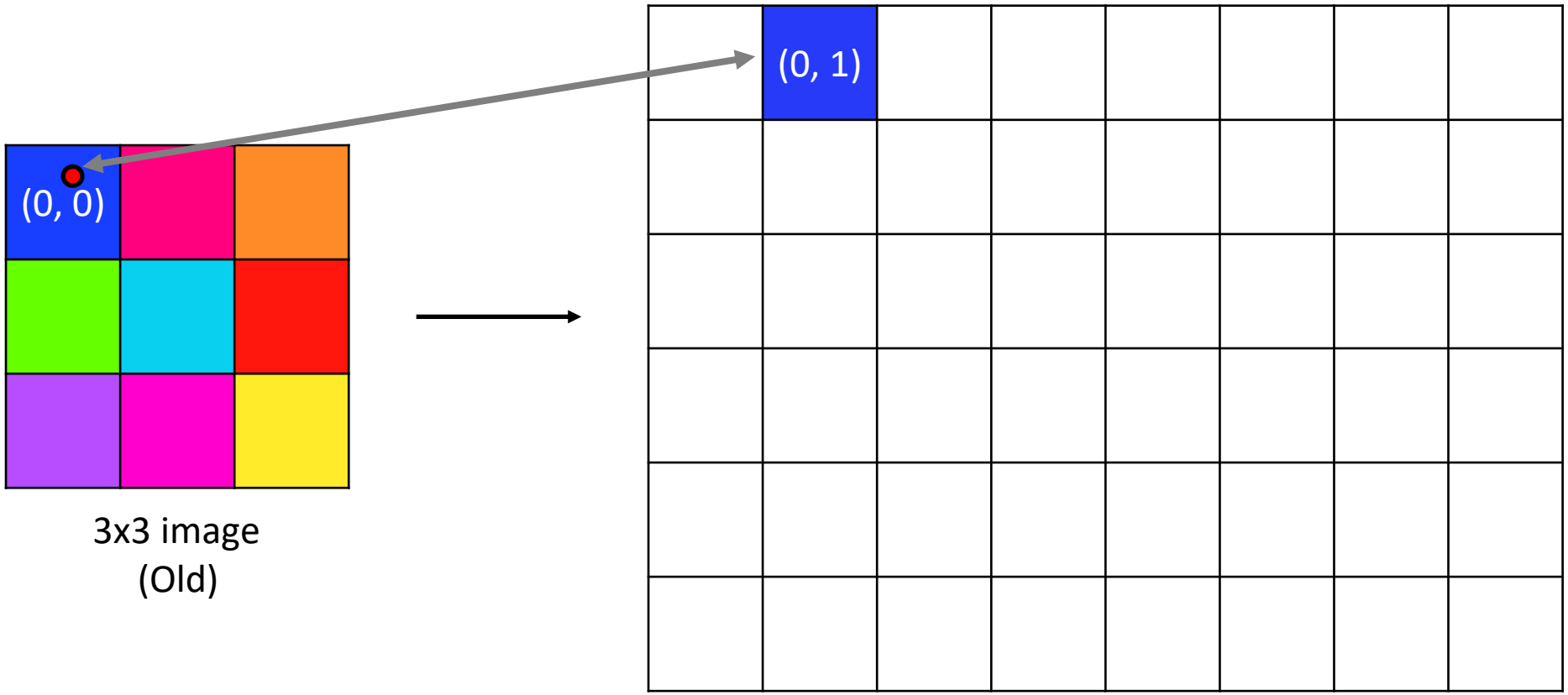


Image resizing

- **Resize 3x3 → 6x8**
  2. Match up coordinates

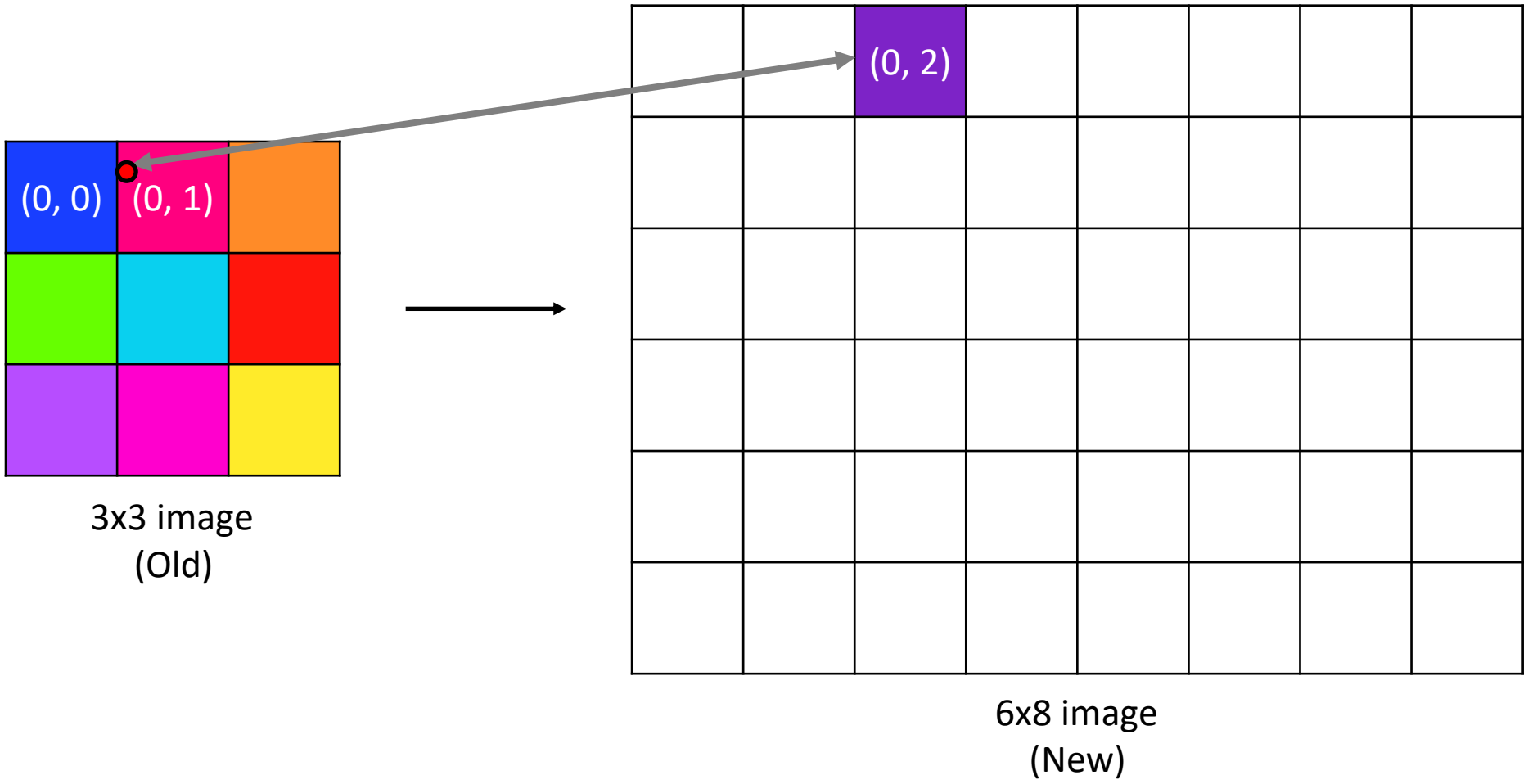


Image resizing

- **Resize 3x3 → 6x8**
  2. Match up coordinates

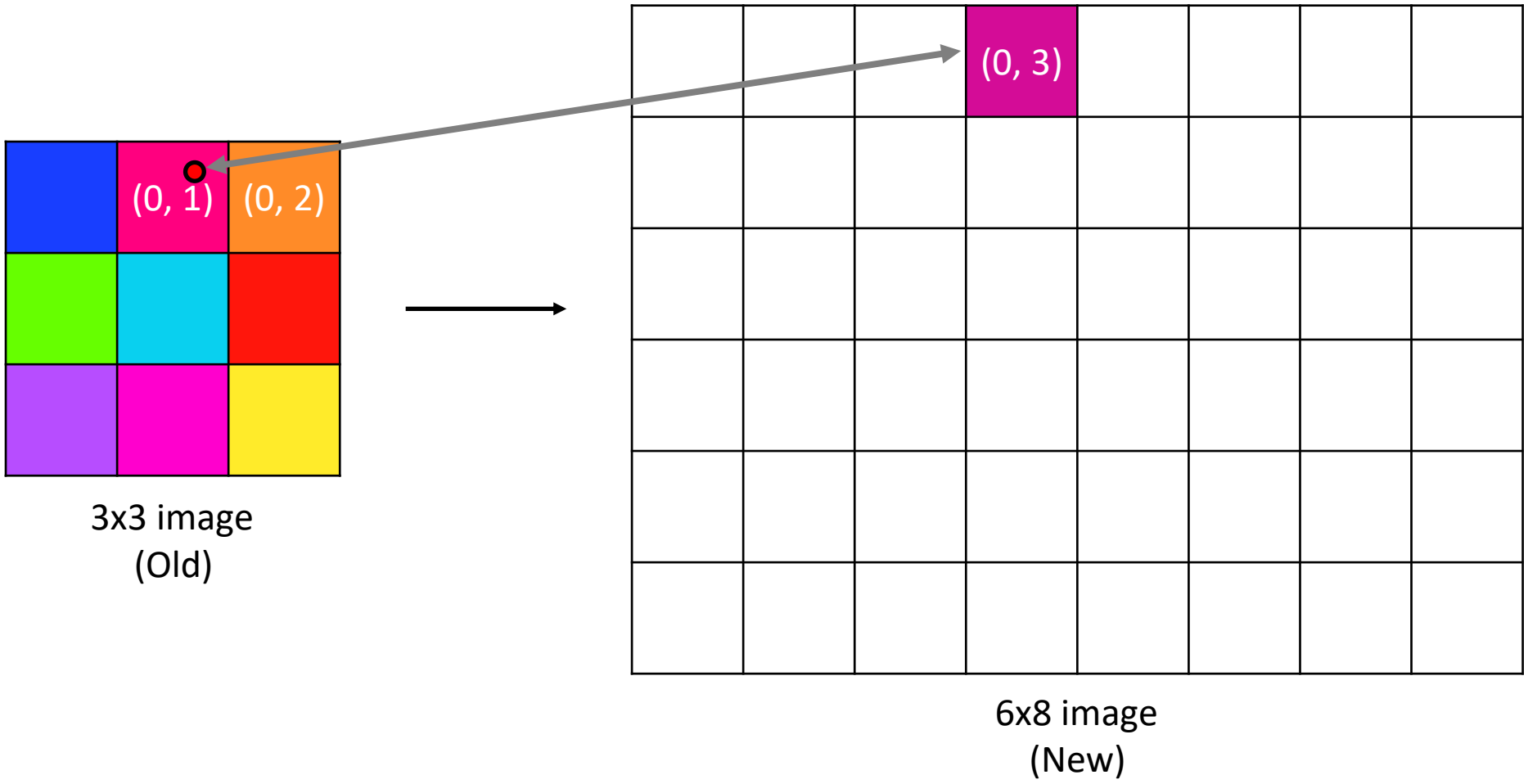


Image resizing

- **Resize 3x3 → 6x8**
  2. Match up coordinates

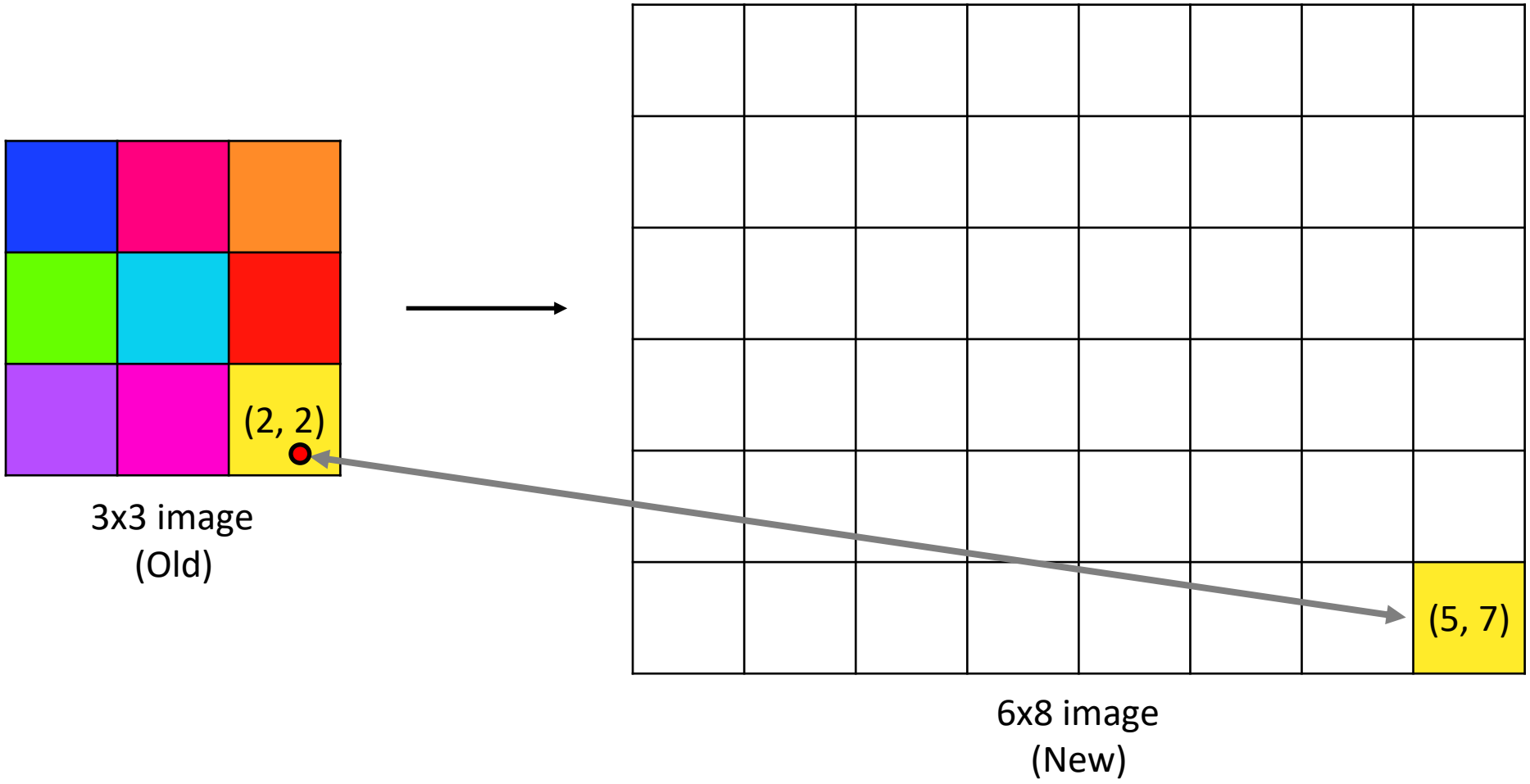
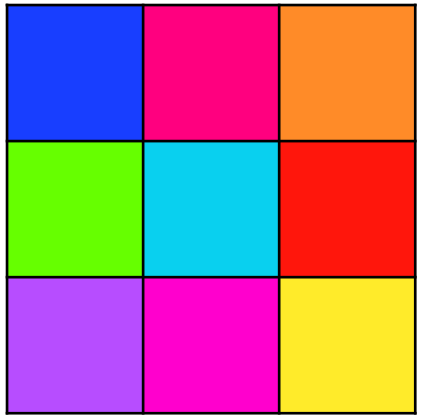
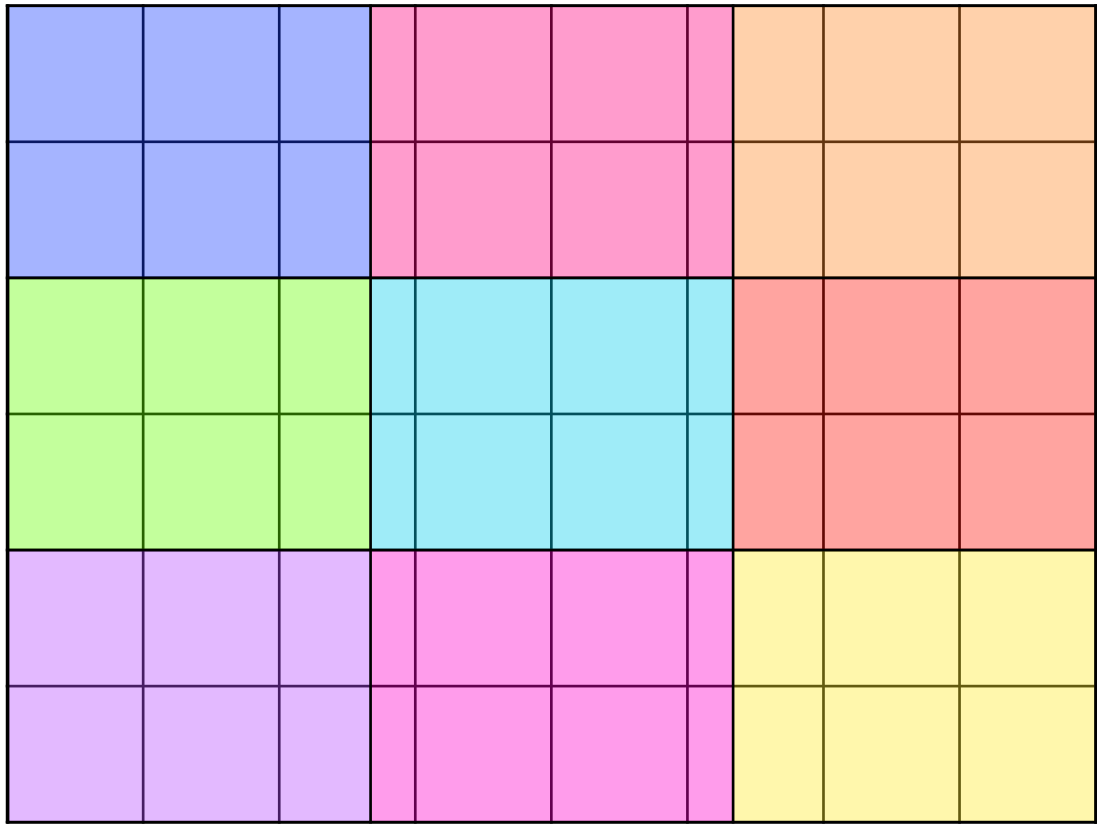


Image resizing

- **Resize 3x3 → 6x8**
  2. Match up coordinates



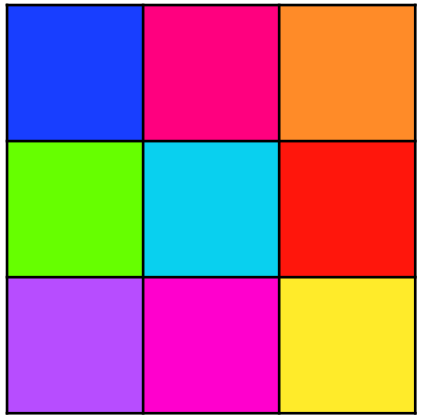
3x3 image
(Old)



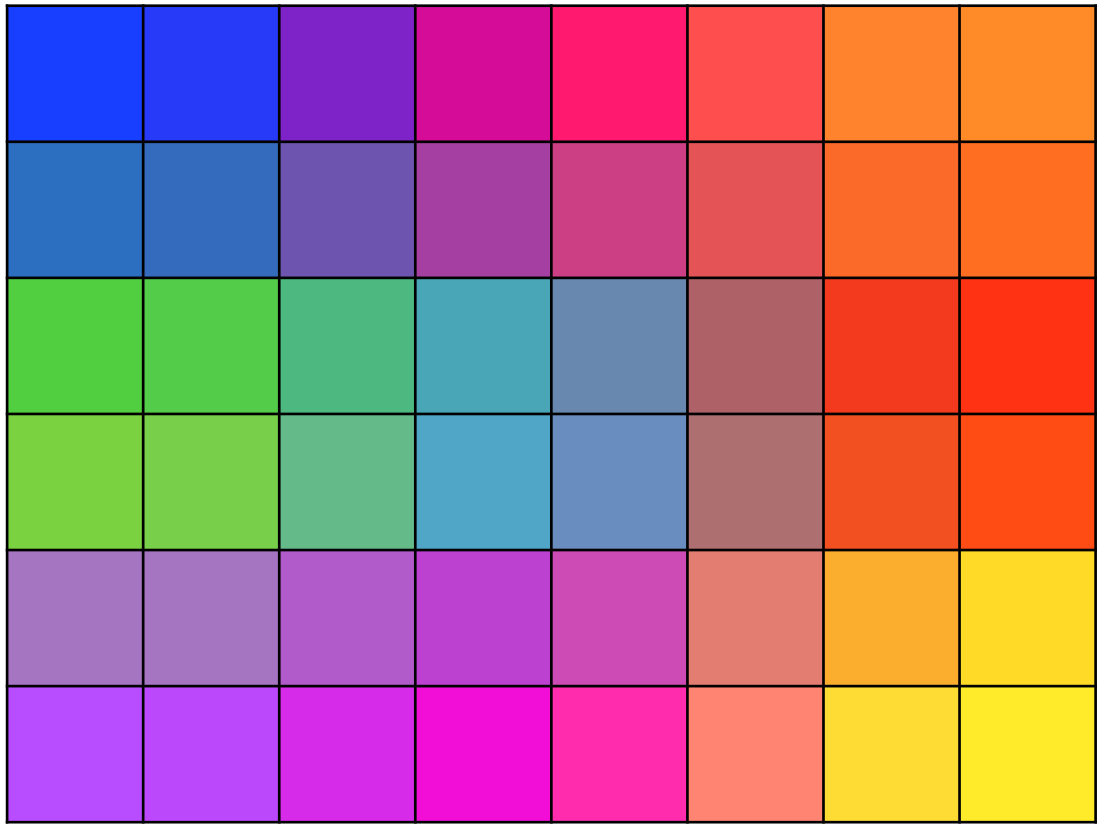
6x8 image
(New)

Image resizing

- **Resize 3x3 → 6x8**
  2. Match up coordinates

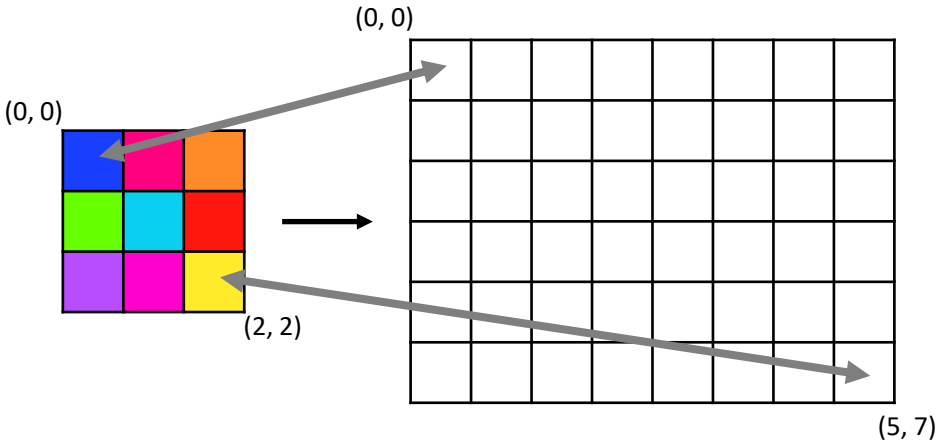


3x3 image
(Old)



6x8 image
(New)

Image resizing



- **Resize 3x3 → 6x8**
  2. Match up coordinates (Pixel: 1x1 Area)
     - $y_{\mathbf{old}} = a_y y_{\mathbf{new}} + b_y$
     - $x_{\mathbf{old}} = a_x x_{\mathbf{new}} + b_x$
       - Old image 좌측 상단 좌표 ↔ New image 좌측 상단 좌표

         $(0,0) \quad \leftrightarrow \quad (0,0)$
       - Old image 우측 하단 좌표 ↔ New image 우측 하단 좌표

         $(2,2) \quad \leftrightarrow \quad (5,7)$

$$\begin{cases} 0 \cdot a_y + b_y = 0 \\ 5 \cdot a_y + b_y = 2' \end{cases} \qquad a_y = \frac{2}{5}, \qquad b_y = 0\,, \qquad \therefore y_{\mathbf{old}} = \frac{2}{5} y_{\mathbf{new}}$$

$$\begin{cases} 0 \cdot a_x + b_x = 0 \\ 7 \cdot a_x + b_x = 2' \end{cases} \qquad a_x = \frac{2}{7}, \qquad b_x = 0\,, \qquad \therefore x_{\mathbf{old}} = \frac{2}{7} x_{\mathbf{new}}$$

Image resizing

- **Resize 3x3 → 6x8**
  3. Iterate over new points
     - $y_{\mathbf{new}} = \frac{2}{5} y_{\mathbf{old}}$
     - $x_{\mathbf{new}} = \frac{2}{7} x_{\mathbf{old}}$
       - Mapping new coordinate to old coordinate
         1. $(0, 0) \rightarrow (0,\ 0\ )$
         2. $(4, 1) \rightarrow (\frac{8}{5},\ \frac{2}{7}\ )$
         3. $(5, 5) \rightarrow (2,\ \frac{10}{7}\ )$

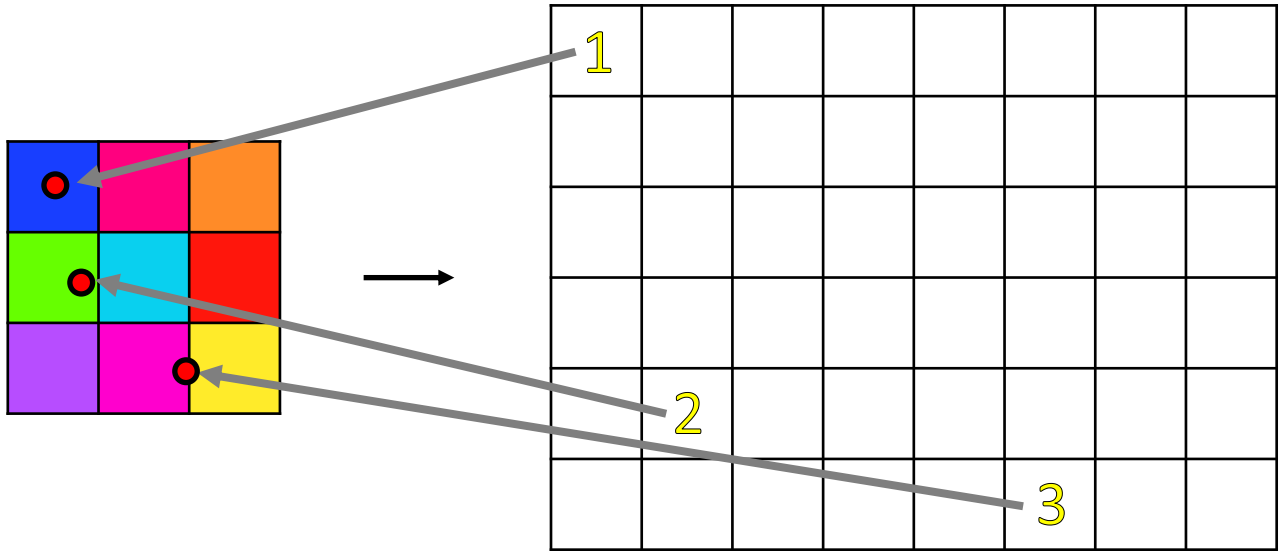


Image resizing

- **Resize 3x3 → 6x8**
  3. Iterate over new points
     - $y_{\mathbf{new}} = \frac{2}{5} y_{\mathbf{old}}$
     - $x_{\mathbf{new}} = \frac{2}{7} x_{\mathbf{old}}$
       - Mapping new coordinate to old coordinate
         1. $(0, 0) \rightarrow (0,\ 0\ )$
         2. $(4, 1) \rightarrow (\frac{8}{5},\ \frac{2}{7}\ )$
         3. $(5, 5) \rightarrow (2,\ \frac{10}{7}\ )$

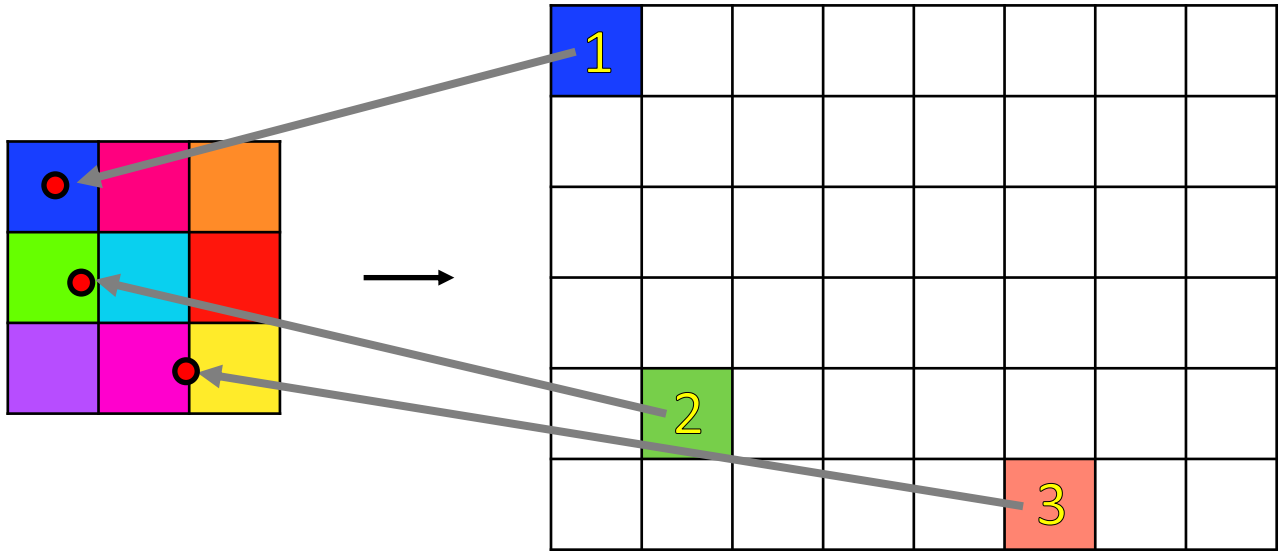
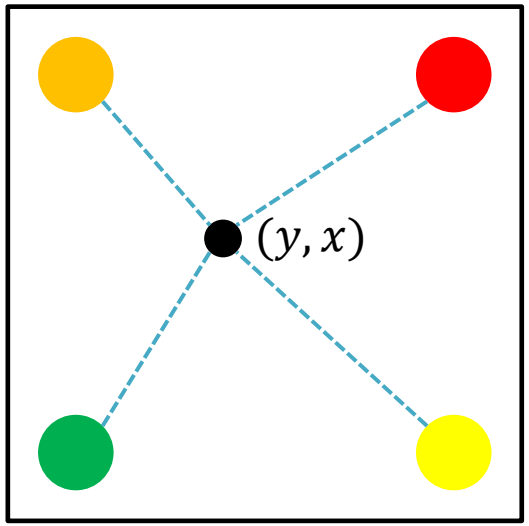
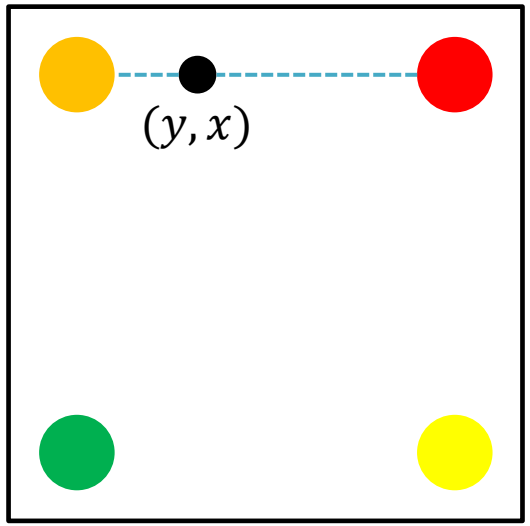


Image resizing

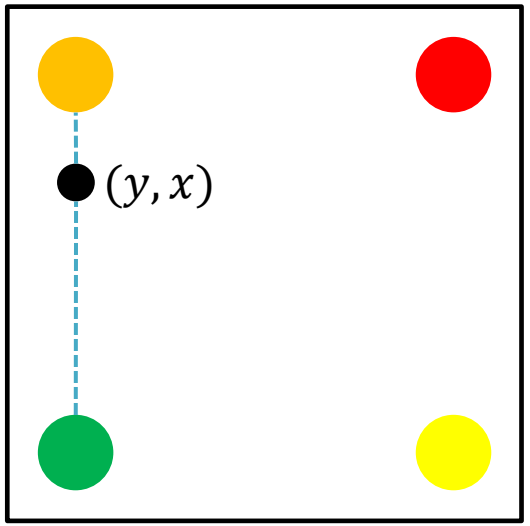
- **Resize 3x3 → 6x8**
  3. Iterate over new points
     - 3 Case



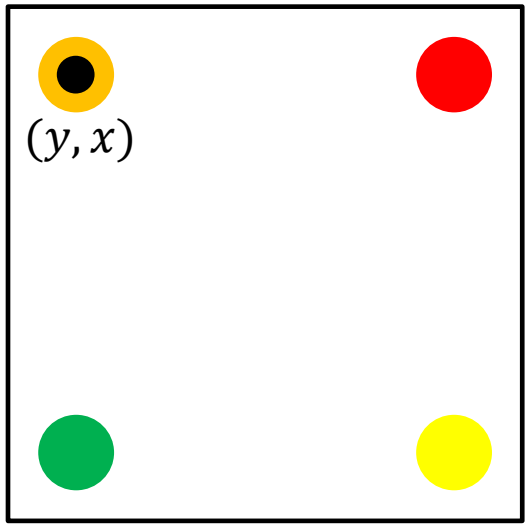
1. 네 점 사이
(float, float)



2-1. 두 점 사이
(int, float)

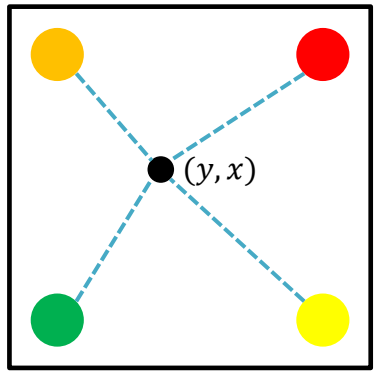


2-2. 두 점 사이
(float, int)



3. 점과 일치
(int, int)

Image resizing



1. 네 점 사이

- **Resize 3x3 → 6x8**

4. Case 1 – Bilinear interpolation

- $f_{\mathbf{new}}(\mathbf{x}_{\mathbf{new}}) = \sum_{i=1}^{4} w_i f_{\mathbf{old}}(\mathbf{x}_i)$
  - Coordinates
    - » $\mathbf{x}_{\mathbf{old}} = (y, x)$
    - » $\mathbf{x}_1 = (\lfloor y \rfloor, \lfloor x \rfloor)$
    - » $\mathbf{x}_2 = (\lfloor y \rfloor, \lfloor x + 1 \rfloor)$
    - » $\mathbf{x}_3 = (\lfloor y + 1 \rfloor, \lfloor x \rfloor)$
    - » $\mathbf{x}_4 = (\lfloor y + 1 \rfloor, \lfloor x + 1 \rfloor)$
  - Weight
    - » $w_1 = (\lfloor y + 1 \rfloor - y)(\lfloor x + 1 \rfloor - x)$
    - » $w_2 = (\lfloor y + 1 \rfloor - y)(x - \lfloor x \rfloor)$
    - » $w_3 = (y - \lfloor y \rfloor)(\lfloor x + 1 \rfloor - x)$
    - » $w_4 = (y - \lfloor y \rfloor)(x - \lfloor x \rfloor)$

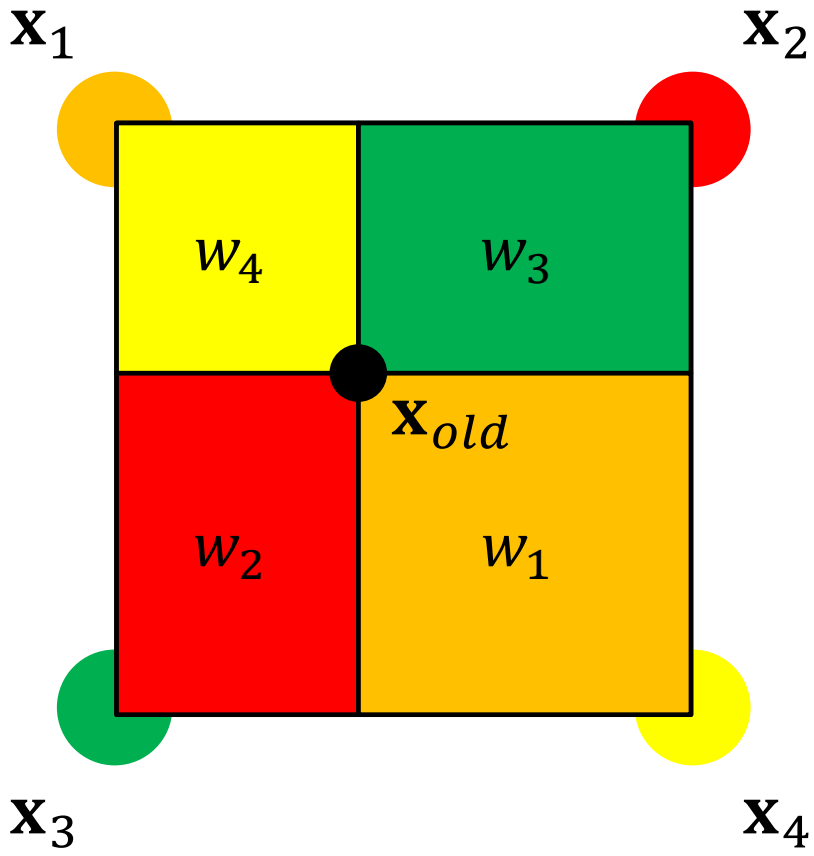
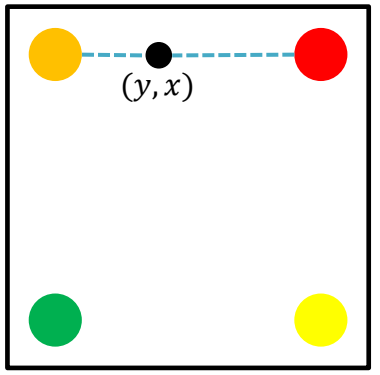


Image resizing



2. 두 점 사이
(int, float)

- **Resize 3x3 → 6x8**
  4. Case 2 – Linear interpolation
     - $f_{\mathbf{new}}(\mathbf{x}_{\mathbf{new}}) = \sum_{i=1}^{2} w_i f_{\mathbf{old}}(\mathbf{x}_i)$
       - Coordinates
         - » $\mathbf{x}_{\mathbf{old}} = (y, x)$
         - » $\mathbf{x}_1 = (y, \lfloor x \rfloor)$
         - » $\mathbf{x}_2 = (y, \lfloor x + 1 \rfloor)$
       - Weight
         - » $w_1 = y(\lfloor x + 1 \rfloor - x)$
         - » $w_2 = y(x - \lfloor x \rfloor)$

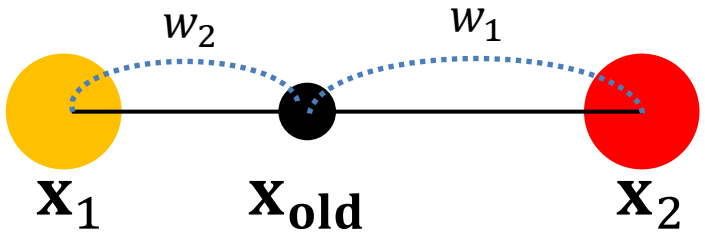
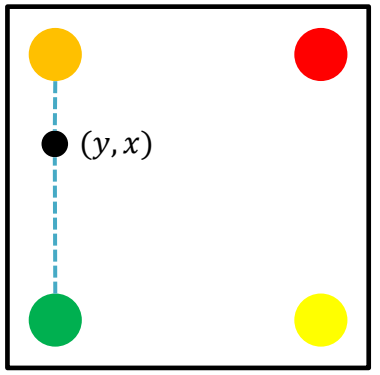


Image resizing



2. 두 점 사이
(float, int)

- **Resize 3x3 → 6x8**

4. Case 2 – Linear interpolation
   - $f_{\mathbf{new}}(\mathbf{x}_{\mathbf{new}}) = \sum_{i=1}^{2} w_i f_{\mathbf{old}}(\mathbf{x}_i)$
     - Coordinates
       - » $\mathbf{x}_{\mathbf{old}} = (y, x)$
       - » $\mathbf{x}_1 = (\lfloor y \rfloor, x)$
       - » $\mathbf{x}_2 = (\lfloor y + 1 \rfloor, x)$
     - Weight
       - » $w_1 = x(\lfloor y + 1 \rfloor - y)$
       - » $w_2 = x(y - \lfloor y \rfloor)$

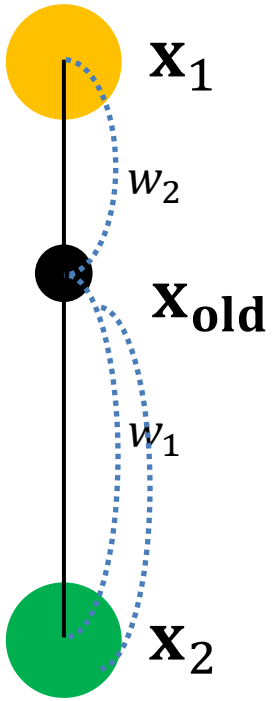
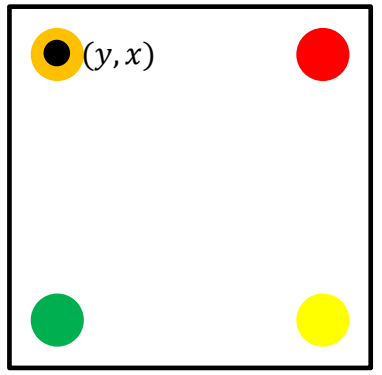


Image resizing



3. 점과 일치
(int, int)

- **Resize 3x3 → 6x8**
  4. Case 3
     - $f_{\mathbf{new}}(\mathbf{x}_{\mathbf{new}}) = f_{\mathbf{old}}(\mathbf{x}_{\mathbf{old}})$

Image resizing

- **Bilinear interpolation**
  - $f_{\mathbf{new}}(\mathbf{x}_{\mathbf{new}}) = \sum_{i=1}^{4} w_i f_{\mathbf{old}}(\mathbf{x}_i)$
    - Coordinates
      - $\mathbf{x}_{\mathbf{old}} = (y, x)$
      - $\mathbf{x}_1 = (\lfloor y \rfloor, \lfloor x \rfloor)$
      - $\mathbf{x}_2 = (\lfloor y \rfloor, \lfloor x + 1 \rfloor)$
      - $\mathbf{x}_3 = (\lfloor y + 1 \rfloor, \lfloor x \rfloor)$
      - $\mathbf{x}_4 = (\lfloor y + 1 \rfloor, \lfloor x + 1 \rfloor)$
    - Weight
      - $w_1 = (\lfloor y + 1 \rfloor - y)(\lfloor x + 1 \rfloor - x)$
      - $w_2 = (\lfloor y + 1 \rfloor - y)(x - \lfloor x \rfloor)$
      - $w_3 = (y - \lfloor y \rfloor)(\lfloor x + 1 \rfloor - x)$
      - $w_4 = (y - \lfloor y \rfloor)(x - \lfloor x \rfloor)$

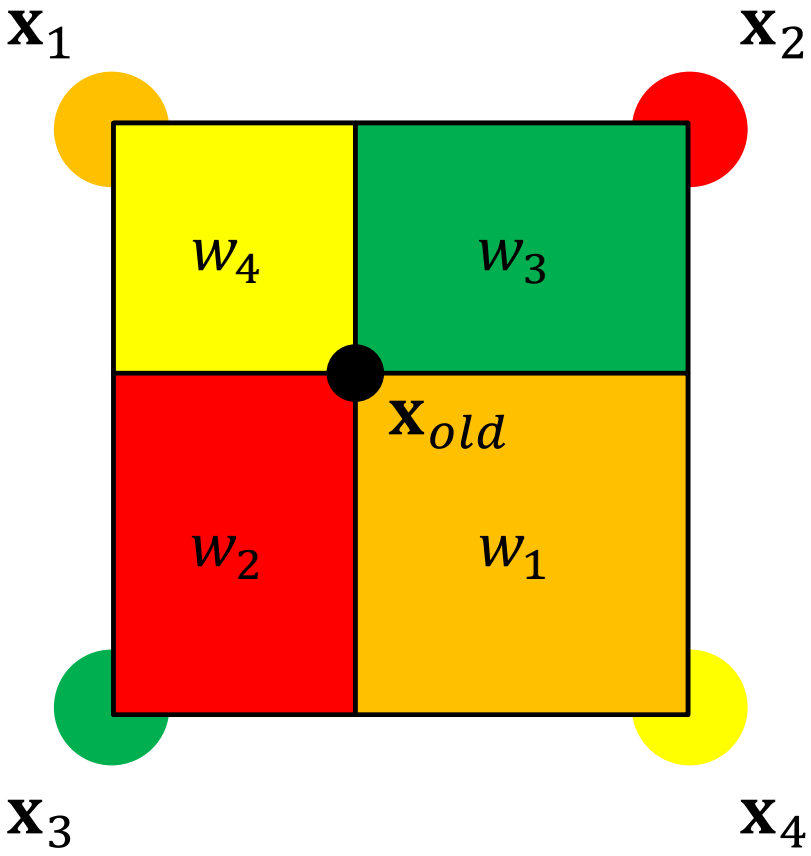


Image resizing

- **Bilinear interpolation**
  - $f_{\mathbf{new}}(\mathbf{x}_{\mathbf{new}}) = \sum_{i=1}^{4} w_i f_{\mathbf{old}}(\mathbf{x}_i)$
    - Coordinates
      - $\mathbf{x}_{\mathbf{old}} = (y, x)$
      - $\mathbf{x}_1 = (\lfloor y \rfloor, \lfloor x \rfloor)$
      - $\mathbf{x}_2 = (\lfloor y \rfloor, \lfloor x + 1 \rfloor)$
      - $\mathbf{x}_3 = (\lfloor y + 1 \rfloor, \lfloor x \rfloor)$
      - $\mathbf{x}_4 = (\lfloor y + 1 \rfloor, \lfloor x + 1 \rfloor)$
    - Weight
      - $w_1 = (\lfloor y + 1 \rfloor - y)(\lfloor x + 1 \rfloor - x)$
      - $w_2 = (\lfloor y + 1 \rfloor - y)(x - \lfloor x \rfloor)$
      - $w_3 = (y - \lfloor y \rfloor)(\lfloor x + 1 \rfloor - x)$
      - $w_4 = (y - \lfloor y \rfloor)(x - \lfloor x \rfloor)$

$$\mathrm{y} = \lfloor y \rfloor \quad \therefore (y - \lfloor y \rfloor) = 0$$

$$\therefore w_3, w_4 = 0$$

$$\therefore f_{\mathbf{new}}(\mathbf{x}_{\mathbf{new}}) = \sum_{i=1}^{2} w_i f_{\mathbf{old}}(\mathbf{x}_i)$$

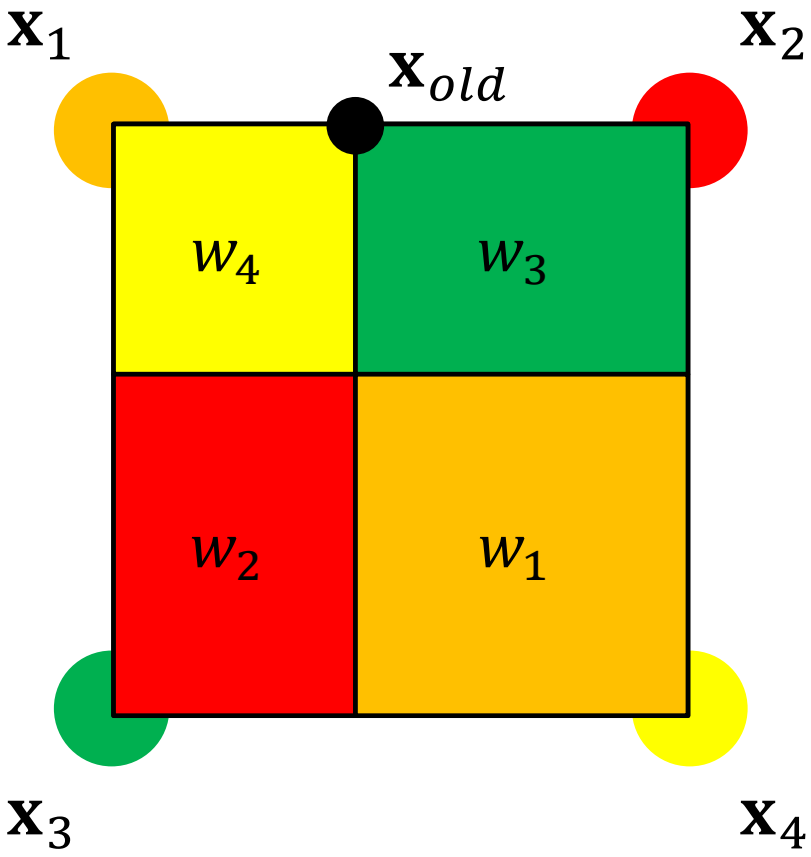


Image resizing

- **Bilinear interpolation**
  - $f_{\mathbf{new}}(\mathbf{x}_{\mathbf{new}}) = \sum_{i=1}^{4} w_i f_{\mathbf{old}}(\mathbf{x}_i)$
    - Coordinates
      - $\mathbf{x}_{\mathbf{old}} = (y, x)$
      - $\mathbf{x}_1 = (\lfloor y \rfloor, \lfloor x \rfloor)$
      - $\mathbf{x}_2 = (\lfloor y \rfloor, \lfloor x + 1 \rfloor)$
      - $\mathbf{x}_3 = (\lfloor y + 1 \rfloor, \lfloor x \rfloor)$
      - $\mathbf{x}_4 = (\lfloor y + 1 \rfloor, \lfloor x + 1 \rfloor)$
    - Weight
      - $w_1 = (\lfloor y + 1 \rfloor - y)(\lfloor x + 1 \rfloor - x)$
      - $w_2 = (\lfloor y + 1 \rfloor - y)(x - \lfloor x \rfloor)$
      - $w_3 = (y - \lfloor y \rfloor)(\lfloor x + 1 \rfloor - x)$
      - $w_4 = (y - \lfloor y \rfloor)(x - \lfloor x \rfloor)$

$$\mathrm{y} = \lfloor y \rfloor, \mathrm{x} = \lfloor x \rfloor \therefore (y - \lfloor y \rfloor) = 0, (x - \lfloor x \rfloor) = 0$$
$$\therefore w_2, w_3, w_4 = 0$$
$$\therefore f_{\mathbf{new}}\,(\mathbf{x}_{\mathbf{new}}) = w_1 f_{\mathbf{old}}(\mathbf{x}_1)$$

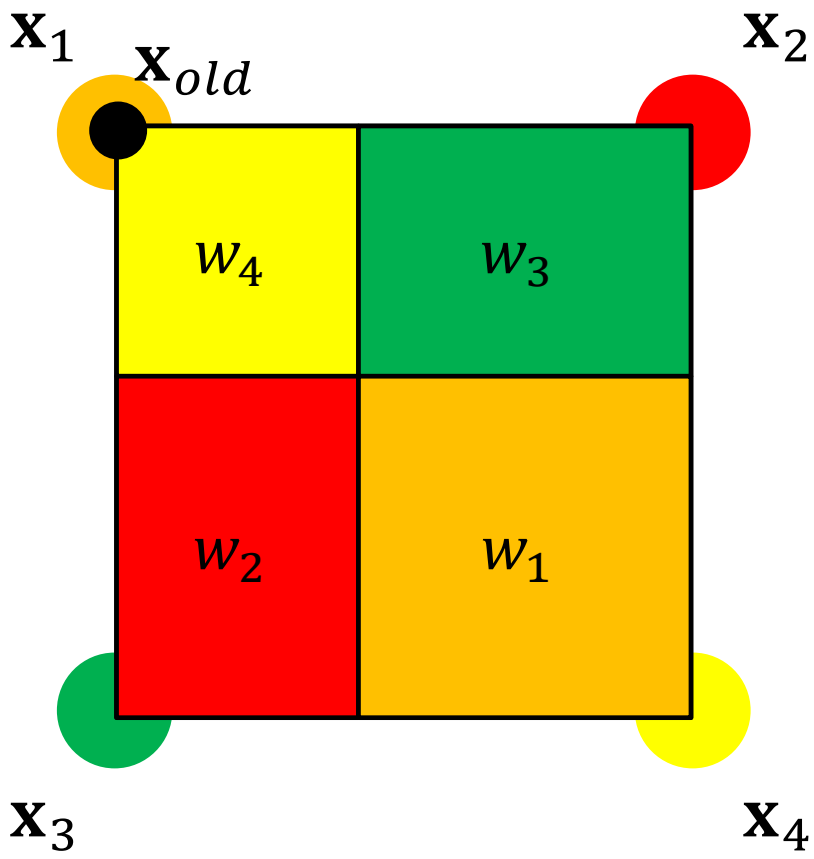


Image resizing

- **Bilinear interpolation**
  - $f_{\mathbf{new}}(\mathbf{x}_{\mathbf{new}}) = \sum_{i=1}^{4} w_i f_{\mathbf{old}}(\mathbf{x}_i)$
    - Coordinates
      - $\mathbf{x}_{\mathbf{old}} = (y, x)$
      - $\mathbf{x}_1 = (\lfloor y \rfloor, \lfloor x \rfloor)$
      - $\mathbf{x}_2 = (\lfloor y \rfloor, \lfloor x + 1 \rfloor)$
      - $\mathbf{x}_3 = (\lfloor y + 1 \rfloor, \lfloor x \rfloor)$
      - $\mathbf{x}_4 = (\lfloor y + 1 \rfloor, \lfloor x + 1 \rfloor)$
    - Weight
      - $w_1 = (\lfloor y + 1 \rfloor - y)(\lfloor x + 1 \rfloor - x)$
      - $w_2 = (\lfloor y + 1 \rfloor - y)(x - \lfloor x \rfloor)$
      - $w_3 = (y - \lfloor y \rfloor)(\lfloor x + 1 \rfloor - x)$
      - $w_4 = (y - \lfloor y \rfloor)(x - \lfloor x \rfloor)$

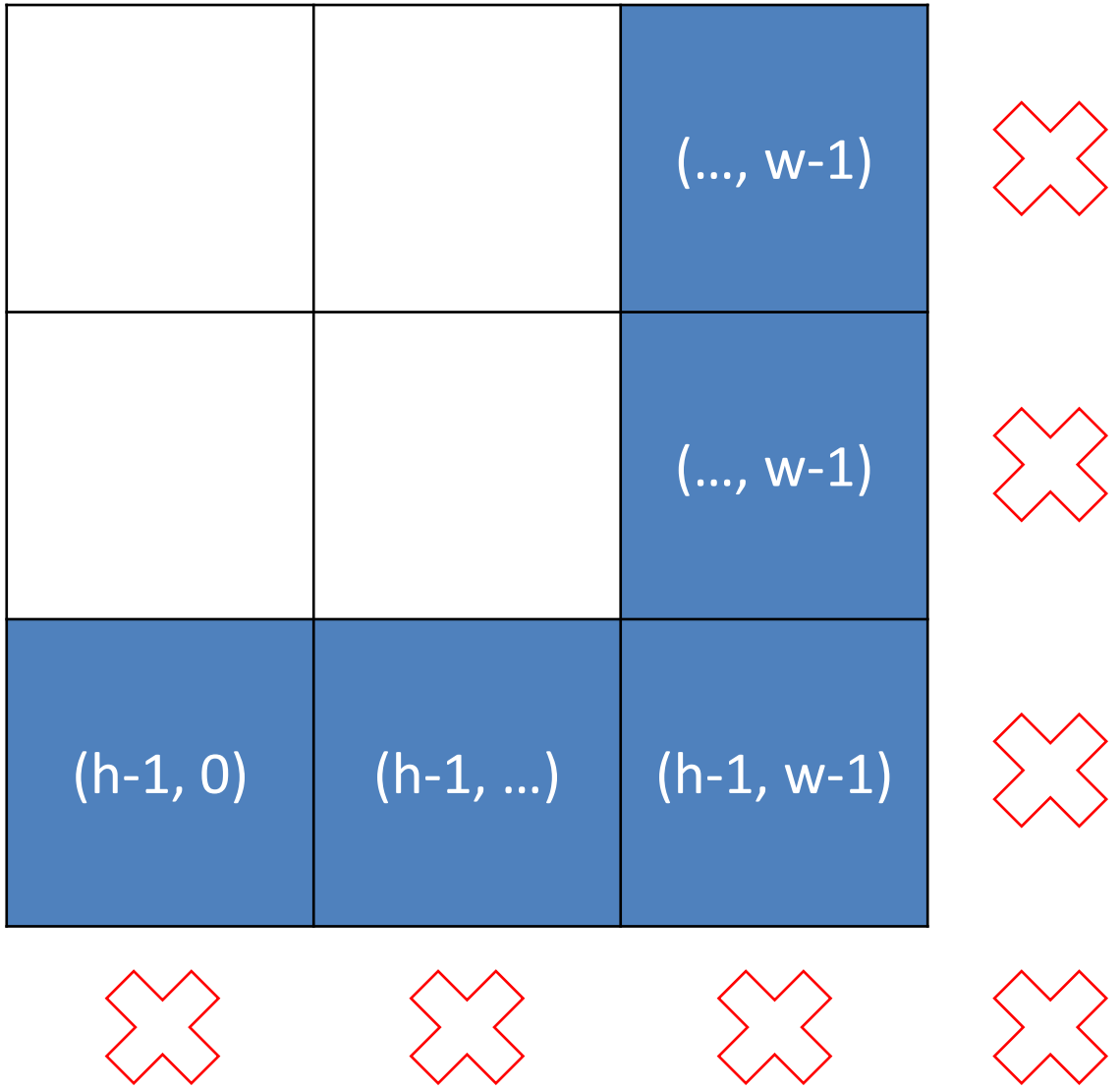
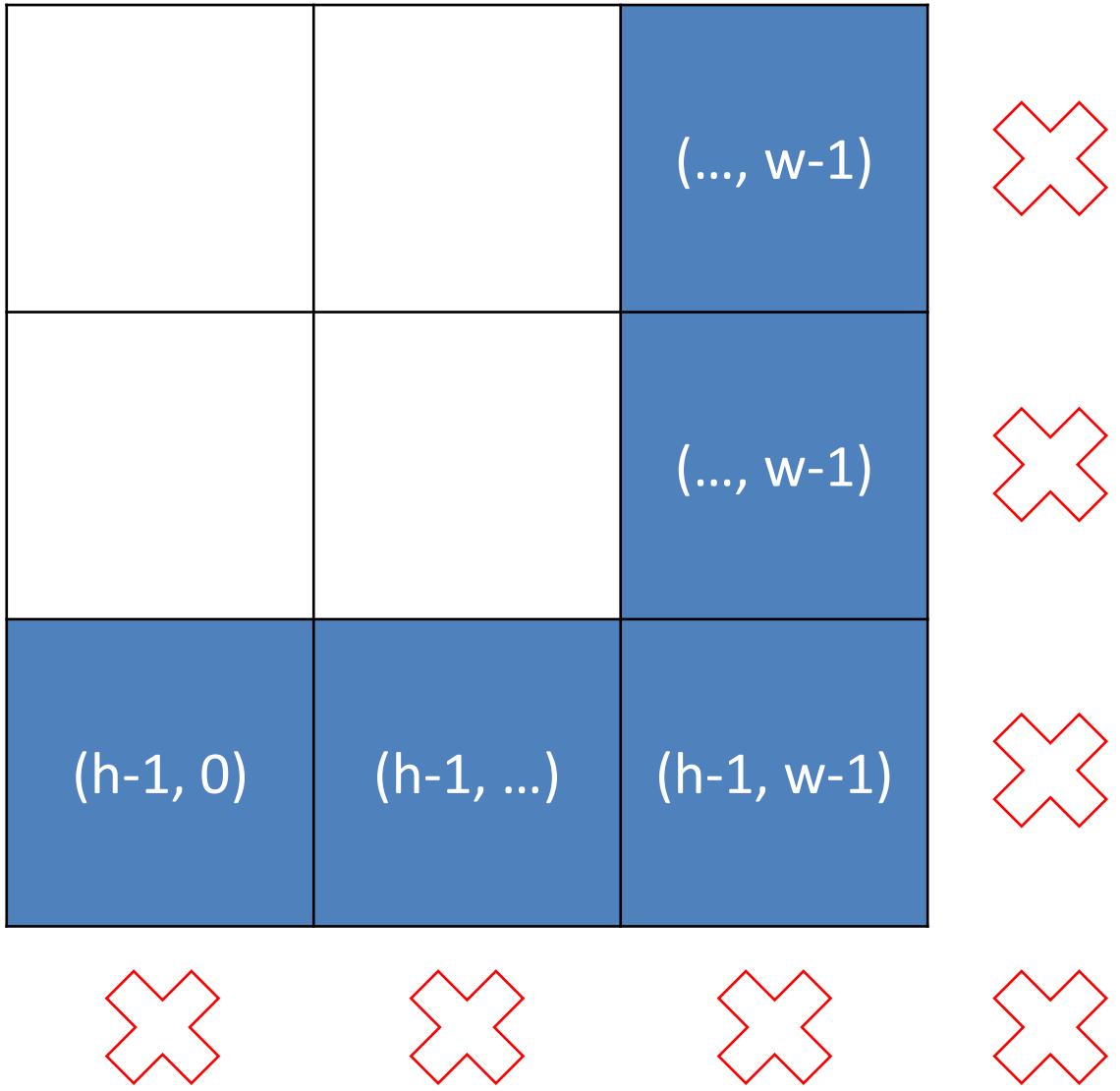


Image resizing

- **Bilinear interpolation**
  - $f_{\mathbf{new}}(\mathbf{x}_{\mathbf{new}}) = \sum_{i=1}^{4} w_i f_{\mathbf{old}}(\mathbf{x}_i)$
    - Coordinates
      - $\mathbf{x}_{\mathbf{old}} = (y, x)$
      - $\mathbf{x}_1 = (\lfloor y \rfloor, \lfloor x \rfloor)$
      - $\mathbf{x}_2 = (\lfloor y \rfloor, \lfloor x + 1 \rfloor)$
      - $\mathbf{x}_3 = (\lfloor y + 1 \rfloor, \lfloor x \rfloor)$
      - $\mathbf{x}_4 = (\lfloor y + 1 \rfloor, \lfloor x + 1 \rfloor)$
    - Weight
      - $w_1 = (\lfloor y + 1 \rfloor - y)(\lfloor x + 1 \rfloor - x)$
      - $w_2 = (\lfloor y + 1 \rfloor - y)(x - \lfloor x \rfloor)$
      - $w_3 = (y - \lfloor y \rfloor)(\lfloor x + 1 \rfloor - x)$
      - $w_4 = (y - \lfloor y \rfloor)(x - \lfloor x \rfloor)$
    - $\lfloor y + 1 \rfloor = \min(\mathrm{h} - 1, \lfloor y + 1 \rfloor)$
    - $\lfloor x + 1 \rfloor = \min(w - 1, \lfloor x + 1 \rfloor)$



과제 my_resize.py

- **main():**
  1. old_img: Lena.png를 grayscale로 읽어옴
  2. new_img_1000by1024: my_resize()함수를 통해 old_img를 (1000, 1024)로 resize
  3. new_img_256by200: my_resize()함수를 통해 old_img를 (256, 200)로 resize
  4. OpenCV를 이용하여 new_img_1000by1024 저장
  5. OpenCV를 이용하여 new_img_256by200 저장

과제 my_resize.py

- **my_resize(old_img, new_shape)**
  - old_img: 불러온 이미지
  - new_shape: new_img의 shape

1. 빈 배열 만들기
2. match_up_coordinates()를 이용하여 a_y, a_x, b_y, b_x를 만들어줌
3. new_img의 모든 픽셀 값을 조회하며 좌표 매칭을 시켜주고 intensity 계산하여 좌표 값 넣어줌

  - return new_img

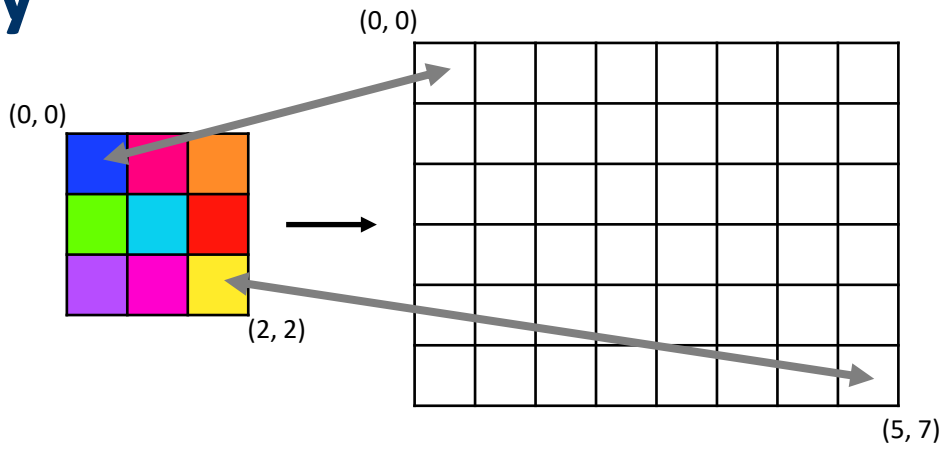
과제 my_resize.py

- **match_up_coordinates(old_shape, new_shape)**
  - old_shape: old_img의 shape
  - new_shape: new_img의 shape

  - return (a_y, b_y, a_x, b_x)

$$y_{\mathbf{old}} = a_y y_{\mathbf{new}} + b_y$$
$$x_{\mathbf{old}} = a_x x_{\mathbf{new}} + b_x$$

과제 my_resize.py



- **match_up_coordinates()**
  - $y_{\mathbf{old}} = a_y y_{\mathbf{new}} + b_y$
  - $x_{\mathbf{old}} = a_x x_{\mathbf{new}} + b_x$
    - Old image 좌측 상단 좌표 ↔ New image 좌측 상단 좌표
      - $(0, 0) \leftrightarrow (0, 0)$
    - Old image 우측 하단 좌표 ↔ New image 우측 하단 좌표
      - $(h_{\mathbf{old}} - 1, w_{\mathbf{old}} - 1) \leftrightarrow (h_{\mathbf{new}} - 1, w_{\mathbf{new}} - 1)$

$$\begin{cases} b_y = 0 \\ (h_{\mathbf{new}} - 1)a_y + b_y = h_{\mathbf{old}} - 1 \end{cases}$$

$$a_y = \frac{h_{\mathbf{old}} - 1}{h_{\mathbf{new}} - 1}, \qquad b_y = 0$$

$$a_x = \frac{w_{\mathbf{old}} - 1}{w_{\mathbf{new}} - 1}, \qquad b_x = 0$$

과제

- **보고서**
  - 내용
    - 학과, 학번, 이름
    - 구현 코드: 구현한 코드에 대한 간단한 설명
    - 느낀 점: 구현 결과를 보고 느낀 점, 혹은 어려운 점 등
    - 과제 난이도: 개인적으로 느낀 난이도 및 이유(과제가 쉽다, 어렵다 등)
  - .pdf 파일로 제출(이외의 파일 형식일 경우 감점)
  - 보고서 명
    - [IP]20xxxxxxx_이름_x주차_과제.pdf
- **이미지:**
  - new_img_1000by1024.png (너비가 1024, 높이가 1000)
  - new_img_256by200.png (너비가 200, 높이가 256)

과제

- **과제 안내**
  - 채점 기준
    - 구현을 못하거나 잘못 구현한 경우
    - 보고서 내용이 빠진 경우
    - 다른 사람의 코드 copy 적발시 보여준 사람, copy한 사람 둘 다 0점
    - 내장 함수 사용시 감점(내장 함수를 사용해도 된다고 한 것 제외)
  - 제출 파일
    - 아래의 파일을 압축해서 [IP]20XXXXXXX_이름_X주차_과제.zip 으로 제출
      - .py 파일
      - .pdf 보고서 파일
      - .png 파일
  - 제출 기한
    - 2024년 4월 18일 23시 59분까지

Q & A