Augmented reality: Lesson 2
Image resizing and interpolation

Francesco Di Giacomo

Augmented reality course
Francesco Di Giacomo
giacf@hr.nl

- Given an input picture, change the size of the picture and output the new image.
- The new image is *scaled*: larger or smaller.
- Usually we scale the width and the height by the same amount to preserve aspect ratio.
- **Application in this course:** Image segmentation, omnidirectional cameras.

- Image size: 640x480.
- We double the size of the image.
- Amount of pixels in the original image: 294400.
- Amount of pixels in the scaled image: 1228800.
- Some pixels match pixels in the image.
- What about the extra pixels? What do we do with them?

Image resizing

An example

- Image size: 640x480.
- We double the size of the image.
- Amount of pixels in the original image: 294400.
- Amount of pixels in the scaled image: 1228800.
- Some pixels match pixels in the image.
- What about the extra pixels? What do we do with them?

- We have to find where the pixel was in the original image, given the scaled image position.
- We write the scale factor on the height as s_r and the scale factor on the width as s_c.
- The pixel coordinates in the scaled image are $(r', c') = (s_r \cdot r, s_c \cdot c)$.
- The corresponding pixel coordinates in the original image are $(r, c) = (r'/s_r, c'/s_c)$.

- We scale the image by 1.5. Original size: 320 x 240.
- The pixel $(150, 330)$ in the scaled image correspond to $(150/1.5, 330/1.5) = (100, 220)$ in the original image.
- What about pixel $(76, 20)$? Applying the same transformation gives $(76/1.5, 20/1.5) = (50.67, 13.3)$
- Pixels with decimal coordinates?????!!!

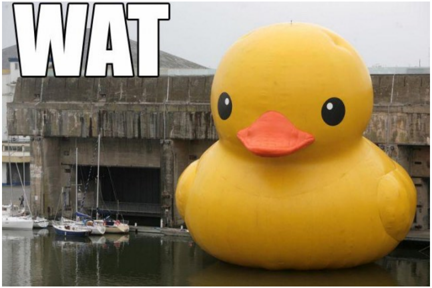
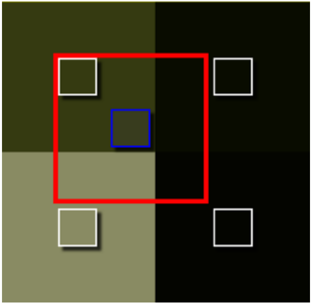


Image resizing

Why are we getting fractional pixels?

- We all know (hopefully!) that given an integer number the division does not always return an integer number.
- This is why we get fractional pixels.
- The reason is that the amount of pixels in the scaled image is different from the amount of pixels in the original image.
- The mapping is not 1:1.

- We cannot directly read the colour values from the starting image.
- Why? Because pixel $(50.67, 13.3)$ does not make any sense (coordinates are integer numbers).
- We have to "guess" the colour components of the fractional pixel, based on non-fractional pixel neighbours in the original image.
- This is where interpolation comes into play.



- We all know (hopefully!) that given an integer number the division does not always return an integer number.
- This is why we get fractional pixels.
- The reason is that the amount of pixels in the scaled image is different from the amount of pixels in the original image.
- The mapping is not 1:1.

- What neighbour can we choose?
- What is the simplest solution you can think of?

- What neighbour can we choose?
- What is the simplest solution you can think of?

The nearest one!

- Fractional pixel $(50.67, 13.3)$ is between $(50, 13)$, $(51, 13)$, $(50, 14)$, and $(51, 14)$.
- Which one is the closest?

- Fractional pixel $(50.67, 13.3)$ is between $(50, 13)$, $(51, 13)$, $(50, 14)$, and $(51, 14)$.
- Which one is the closest?
- The closest one is $(51, 13)$.
- How do we get it?

- Fractional pixel $(50.67, 13.3)$ is between $(50, 13)$, $(51, 13)$, $(50, 14)$, and $(51, 14)$.
- Which one is the closest?
- The closest one is $(51, 13)$.
- How do we get it?
- We round the fractional pixel coordinates (r_f, c_f).
- $(r, c) = (round(r_f), round(c_f))$

- Jagged profiles ("stairway" profile). Not very good quality because the approximation is not gradual.
- Easy to implement.
- Fast.

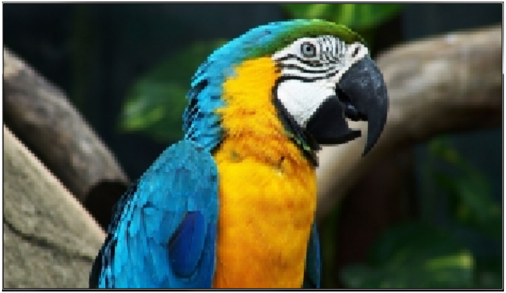


Figure: Image resized with nearest neighbour interpolation

- Improves the quality by adding a gradual approximation of the pixels.
- Harder to implement.
- Slower computation.

Input:

- Original image.
- Scale factors along the rows s_r and the columns s_c of the matrix.
- Again the position of the fractional pixel is $(r_f, c_f) = (r'/s_r, c'/s_c)$ where (r', c') is the position of the pixel in the resized image.

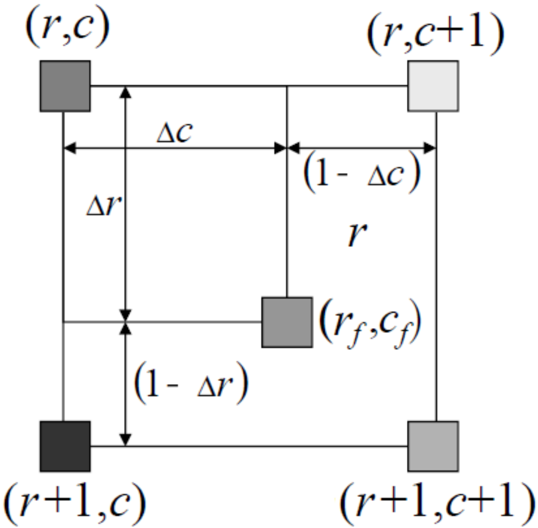
Algorithm:

- Set $r = \lfloor r_f \rfloor$, $c = \lfloor c_f \rfloor$
- Set $\Delta r = r_f - r$ and $\Delta c = c_f - c/$
- What are Δr and Δc?

Algorithm:

- Set $r = \lfloor r_f \rfloor$, $c = \lfloor c_f \rfloor$
- Set $\Delta r = r_f - r$ and $\Delta c = c_f - c/$
- What are Δr and Δc?

Fraction of the distance of the fractional pixel between (r, c) and its 3 neighbours



(r,c)
$(r,c+1)$
Δc
$(1-\Delta c)$
Δr
r
(r_f,c_f)
$(1-\Delta r)$
$(r+1,c)$
$(r+1,c+1)$

Assume we have a greyscale image (only one colour component). Consider the colour component in the original image $I(r, c)$ at coordinates r and c. We compute the new pixel colour component $J(r', c')$ as a weighted sum:

$$
\begin{aligned}
J(r', c') = & I(r, c) \cdot (1 - \Delta r) \cdot (1 - \Delta c) + \\
& I(r + 1, c) \cdot \Delta r \cdot (1 - \Delta c) + \\
& I(r, c + 1) \cdot (1 - \Delta r) \cdot \Delta c + \\
& I(r + 1, c + 1) \cdot \Delta r \cdot \Delta c
\end{aligned}
$$

Example:

We scale the original image I by a factor of 1.5. Consider the pixel at coordinates $(21, 5)$ in the resized image. Assume that the colour components at the four corners are 150, 200, 25, 10 in the original image. We have

$$
\begin{aligned}
&(r_f, c_f) = (20/1.5, 5/1.5) = (13.33, 3.33)\\
&r = \lfloor r_f \rfloor = 13, c = \lfloor c_f \rfloor = 3\\
&\Delta r = r_f - r = 0.33, \Delta c = c_f - c = 0.33\\
J(21,5) =& I(13,3) \cdot 0.77 \cdot 0.77 +\\
&I(14,3) \cdot 0.33 \cdot 0.77 +\\
&I(13,4) \cdot 0.77 \cdot 0.33 +\\
&I(14,4) \cdot 0.33 \cdot 0.33\\
J(21,5) =& 150 \cdot 0.77 \cdot 0.77 +\\
&200 \cdot 0.33 \cdot 0.77 +\\
&25 \cdot 0.77 \cdot 0.33 +\\
&10 \cdot 0.33 \cdot 0.33 \ 147
\end{aligned}
$$

Figure: Image resized with bilinear interpolation

- Apply the bilinear interpolation for the red value.
- Apply the bilinear interpolation for the green value.
- Apply the bilinear interpolation for the blue value.
- Create a new colour containing the interpolated components and place it in the resized image.

Assignment 2

Implement a function which allows you to resize an input image with 3 colour components. The function should be able to both scale up and down an image. Implement the function using both the nearest neighbour and the bilinear interpolation techniques.