# Localized Blurring

- Step 1: Apply Gaussian blurring to the entire image
- Step 2: Compute the final image as a weighted average of the blurred and original image

$$I'(x,y) = w(x,y) * I_b(x,y) + (1 - w(x,y)) * I(x,y)$$

where $I$, $I_b$ and $I'$ are the initial, blurred and final images respectively, $w$ is the weight and $(x,y)$ is the pixel location

# Computing Weights

- Weight should be maximum at the clicked point and decrease on moving away
  - Easiest to use standard distributions
- Gaussian Distribution:

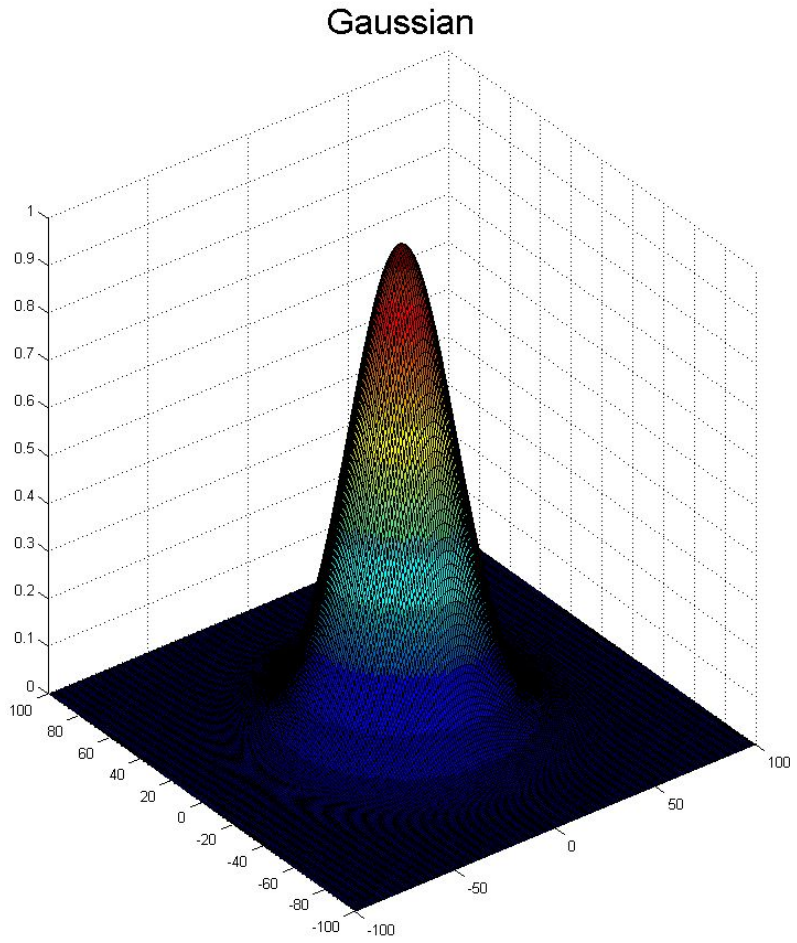
$$w(x,y) = \exp\left(-((x-x_0)^2 + (y-y_0)^2)/\sigma^2\right)$$

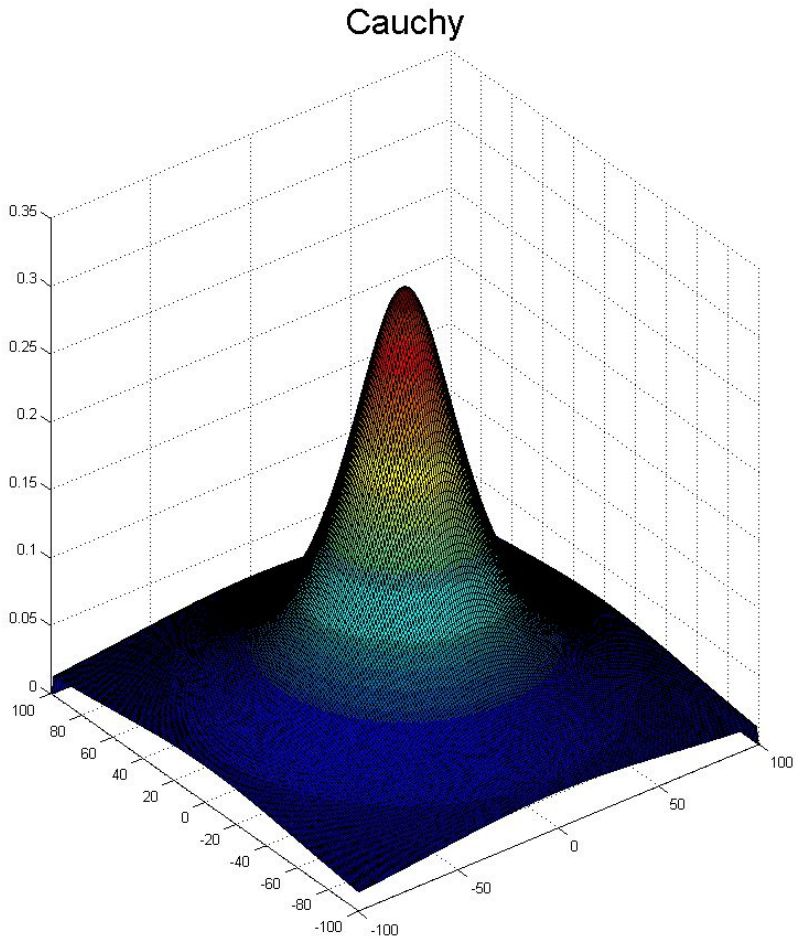
- Cauchy Distribution:

$$w(x,y) = 1/(1 + ((x-x_0)^2 + (y-y_0)^2)/\sigma^2)$$

- where $(x_0, y_0)$ is the clicked point and $\sigma$ is the standard deviation

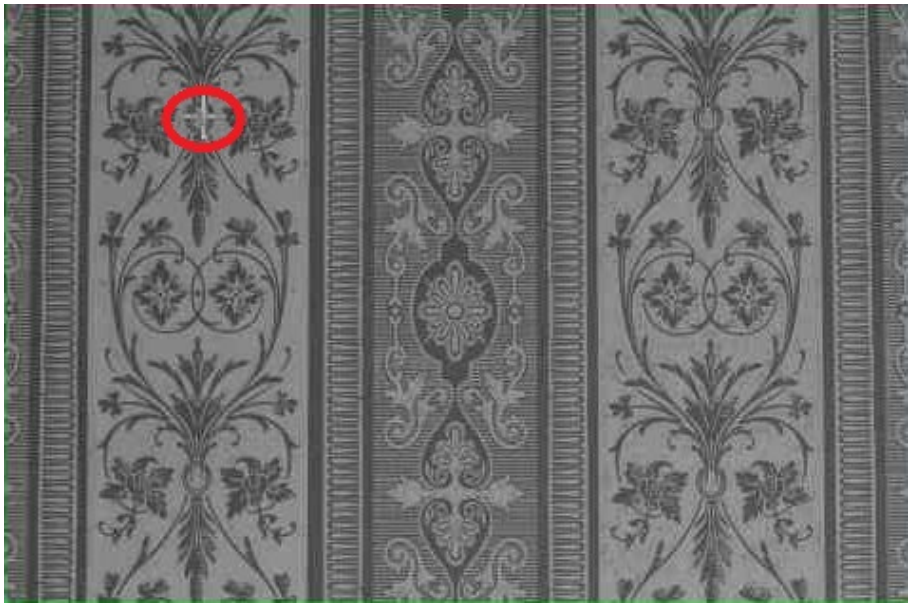
# Distribution Shapes



# Expected Behavior

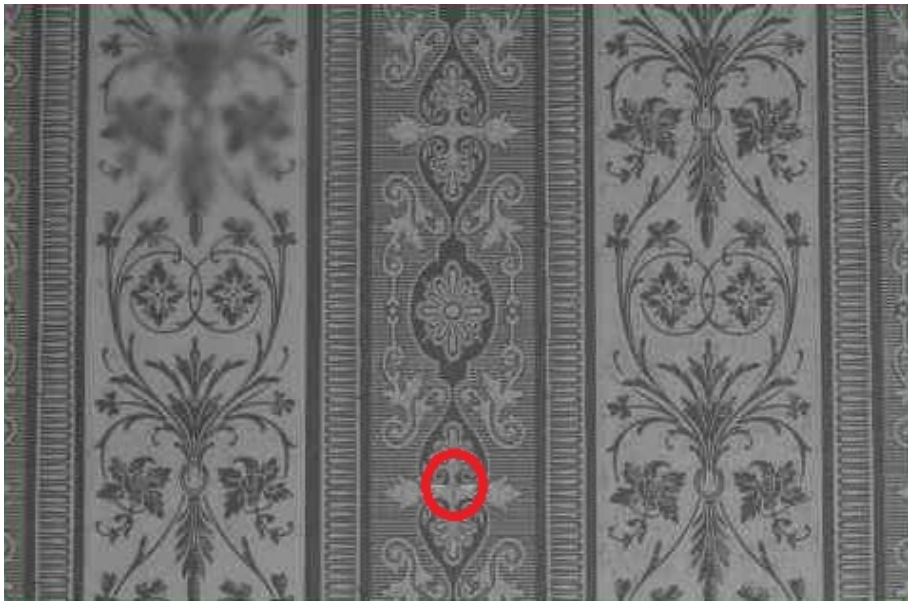
- If we click at this point in the image:



- The result should look like:

# Expected Behavior (cont'd)

- Now if we click at this second point:



- The result should look like:

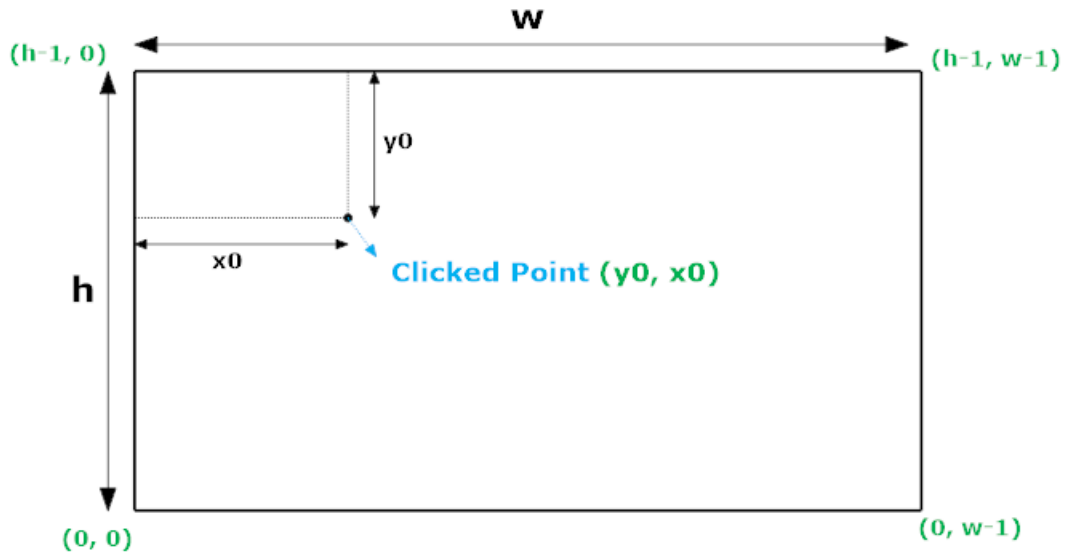
# Implementation Trick

- Naïve method of computing the weight for each pixel is slow and inefficient.
- Better way is to compute the Gaussian (or Cauchy) mask *offline*, i.e. before the loop is started
- On getting a mouse click:
  - **Translate the center of the mask to lie at the clicked point**
  - Compute element wise matrix product between the image and the translated mask

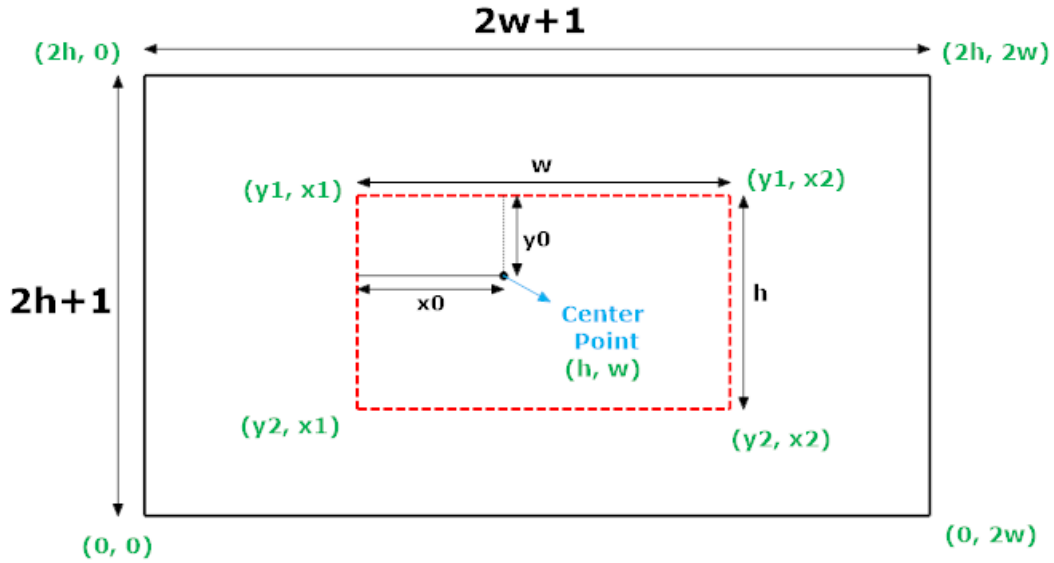
# Translating the Mask

- The mask should be twice as large as the image
  - translated center point remains within its bounds
- Extract a region from the mask
  - Same size as the image
  - Center of the mask is at the same location within this region as the clicked point in the image

# Translating the Mask (cont'd)




Original image with width *w* and height *h* showing the clicked point (point indices in green)




Mask with width *2w+1* and height *2h+1* showing the region to extract with red dashed lines (point indices in green)