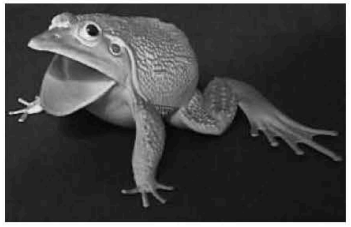
Visual Computing I:

Interactive Computer Graphics and Vision



Features, Features Description and Feature Matching

Stefanie Zollmann and Tobias Langlotz

Last time..

2D Image Processing

- Luminance
  - Brightness
  - Contrast
  - Histogram equalization
  - Gamma
- Color
  - Grayscale
  - Saturation
- Dithering
  - Quantization
  - Random dither
  - Floyd-Steinberg
- Linear filtering
  - Convolution
  - Blur & Sharpen
  - Edge detection
  - Image Pyramids
- Non-linear filtering
  - Median
- Image Feature Detection
  - Edges (e.g Canny Edge)
  - Corners (e.g. Shi-Tomasi, Harris Corner)
  - Blobs (e.g. Difference of Gaussians)

Linear Filters and Image Convolutions

Original

$$\begin{matrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{matrix}$$

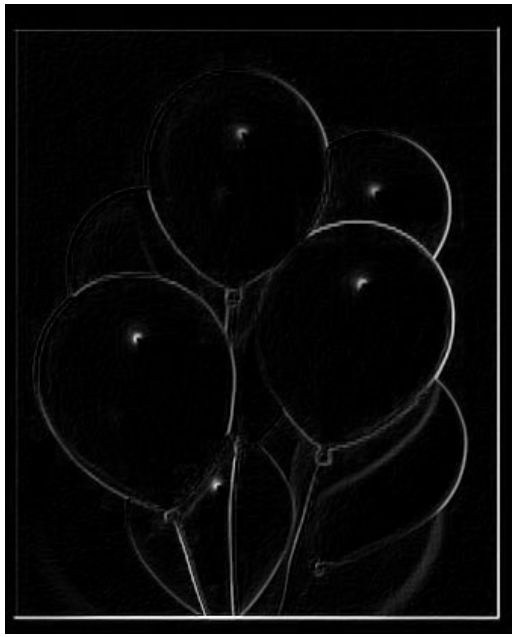
Edge horizontal

$$\begin{matrix} -\frac{1}{2} & 0 & \frac{1}{2} \\ -1 & 0 & 1 \\ -\frac{1}{2} & 0 & \frac{1}{2} \end{matrix}$$

Edge vertical

$$\begin{matrix} -\frac{1}{2} & -1 & -\frac{1}{2} \\ -0 & 0 & 0 \\ \frac{1}{2} & 1 & \frac{1}{2} \end{matrix}$$

Edge (Sobel Operator)



Linear Filters and Image Convolutions

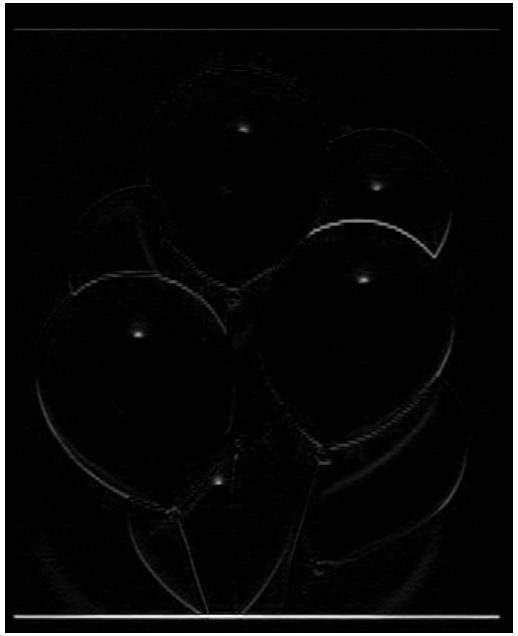
Original

$$\begin{matrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{matrix}$$

Edge horizontal/ vertical

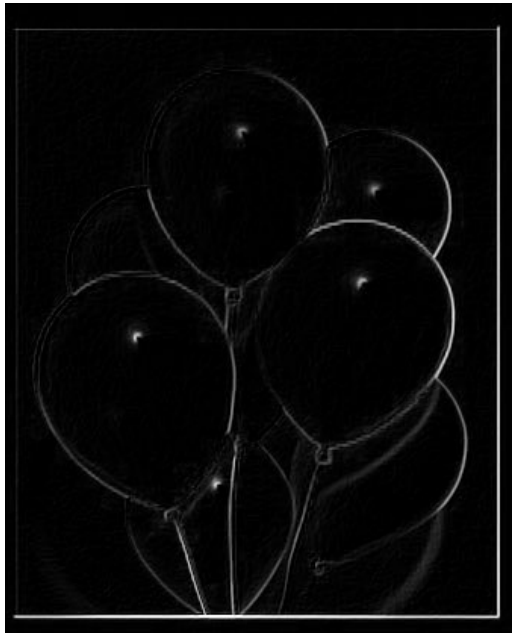
$$\begin{matrix} -\frac{1}{2} & 0 & \frac{1}{2} \\ -1 & 0 & 1 \\ -\frac{1}{2} & 0 & \frac{1}{2} \end{matrix}$$

Edge vertical/ horizontal



$$\begin{matrix} -\frac{1}{2} & -1 & -\frac{1}{2} \\ -0 & 0 & 0 \\ \frac{1}{2} & 1 & \frac{1}{2} \end{matrix}$$

Edge (Sobel Operator)



Linear Filters and Image Convolutions

- Flipping operator in convolutions?!
    - The short answer is: **Yes**, in pure mathematics and traditional signal processing, the definition of convolution involves flipping the kernel. However, also **No** because in computer vision, graphics, and deep learning, the operation we call "convolution" is actually cross-correlation, which does not flip the kernel.
    - There is a lot of subtlety and differences between the meanings of convolution and correlation in different fields (also Tobias did not know about that)

2D Image Processing

- Luminance
  - Brightness
  - Contrast
  - Histogram equalization
  - Gamma
- Color
  - Grayscale
  - Saturation
- Dithering
  - Quantization
  - Random dither
  - Floyd-Steinberg

- Linear filtering
  - Convolution
  - Blur & Sharpen
  - Edge detection
  - Image Pyramids
- Non-linear filtering
  - Median

- Image Feature Detection
  - Edges (e.g Canny Edge)
  - Corners (e.g. Shi-Tomasi, Harris Corner)
  - Blobs (e.g. Difference of Gaussians)

2D Image Processing

- Luminance
  - Brightness
  - Contrast
  - Histogram equalization
  - Gamma
- Color
  - Grayscale
  - Saturation
- Dithering
  - Quantization
  - Random dither
  - Floyd-Steinberg

- Linear filtering
  - Convolution
  - Blur & Sharpen
  - Edge detection
  - Image Pyramids
- Non-linear filtering
  - Median

- Image Feature Detection
  - Edges (e.g Canny Edge)
  - Corners (e.g. Shi-Tomasi, Harris Corner)
  - Blobs (e.g. Difference of Gaussians)

Image Filters - Non-Linear Filters

Non-Linear Filters

- Not all filters are implemented as convolution operations
- E.g. Median filter
  - Take a local neighbourhood (e.g. a 3×3 or 5×5 window)
  - Sort the pixel values
  - Use the middle value as the result
  - Has a smoothing effect, but preserves sharp edges

Original

Median

2D Image Processing

- Luminance
  - Brightness
  - Contrast
  - Histogram equalization
  - Gamma
- Color
  - Grayscale
  - Saturation
- Dithering
  - Quantization
  - Random dither
  - Floyd-Steinberg

- Linear filtering
  - Convolution
  - Blur & Sharpen
  - Edge detection
  - Image Pyramids
- Non-linear filtering
  - Median

- Image Feature Detection
  - Edges (e.g Canny Edge)
  - Corners (e.g. Shi-Tomasi, Harris Corner)
  - Blobs (e.g. Difference of Gaussians)

2D Image Processing

- Luminance
  - Brightness
  - Contrast
  - Histogram equalization
  - Gamma
- Color
  - Grayscale
  - Saturation
- Dithering
  - Quantization
  - Random dither
  - Floyd-Steinberg

- Linear filtering
  - Convolution
  - Blur & Sharpen
  - Edge detection
  - Image Pyramids
- Non-linear filtering
  - Median

- Image Feature Detection
  - Edges (e.g Canny Edge)
  - Corners (e.g. Shi-Tomasi, Harris Corner)
  - Blobs (e.g. Difference of Gaussians)

2D Image Processing

- Luminance
  - Brightness
  - Contrast
  - Histogram equalization
  - Gamma
- Color
  - Grayscale
  - Saturation
- Dithering
  - Quantization
  - Random dither
  - Floyd-Steinberg

- Linear filtering
  - Convolution
  - Blur & Sharpen
  - Edge detection
  - Image Pyramids
- Non-linear filtering
  - Median

- Image Feature Detection
  - Edges (e.g Canny Edge)
  - Corners (e.g. Shi-Tomasi, Harris Corner)
  - Blobs (e.g. Difference of Gaussians)

Image Features & Feature Detection

Image Features & Feature Detection

- Often required:
    - Features in an image that can be reliably detected in different images of the same scene
- A ‘feature’ or ‘key point’ is
    - A 2D location in an image
    - Accurately/precisely located
    - Can be found in different views

What are good image features?

Image Features & Feature Detection

- Often required:
    - Features in an image that can be reliably detected in different images of the same scene
- A ‘feature’ or ‘key point’ is
    - A 2D location in an image
    - Accurately/precisely located
    - Can be found in different views
- Feature descriptors
    - Used to tell features apart
    - Ideally robust to changes in lighting, viewpoint, contrast, etc.
- Three common types of features
    - Edges – gradient-based
    - Corners – gradient-based
    - Blobs – bright/dark centre/surround

What are good image features?

Image Features & Feature Detection

- Often required:
    - Features in an image that can be reliably detected in different images of the same scene
- A ‘feature’ or ‘key point’ is
    - A 2D location in an image
    - Accurately/precisely located
    - Can be found in different views
- Feature descriptors
    - Used to tell features apart
    - Ideally robust to changes in lighting, viewpoint, contrast, etc.
- Three common types of features
    - Edges – gradient-based
    - Corners – gradient-based
    - Blobs – bright/dark centre/surround

Example: Corners

Image Features & Feature Detection

- Often required:
    - Features in an image that can be reliably detected in different images of the same scene
- A ‘feature’ or ‘key point’ is
    - A 2D location in an image
    - Accurately/precisely located
    - Can be found in different views
- Feature descriptors
    - Used to tell features apart
    - Ideally robust to changes in lighting, viewpoint, contrast, etc.
- Three common types of features
    - Edges – gradient-based
    - Corners – gradient-based
    - Blobs – bright/dark centre/surround

Example: Blobs

Edge Detection - Canny Edge Detector

- Canny Edge Detector
  - Step 1: Reduce noise (> Gaussian filter G_σ)

Original

Edge Detection - Canny Edge Detector

- Canny Edge Detector
  - Step 1: Reduce noise (> Gaussian filter G_σ)
  - Step 2: Compute gradients in x and y direction (> Sobel operators H_u, H_v), gradient is a vector $\mathbf{g} = \begin{bmatrix} H_u & H_v \end{bmatrix}^T$

Original

$$H_u = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad H_v = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Edge Detection - Canny Edge Detector

- Canny Edge Detector
  - Step 1: Reduce noise (> Gaussian filter G_σ)
  - Step 2: Compute gradients in x and y direction (> Sobel operators H_u, H_v), gradient is a vector $\mathbf{g} = \begin{bmatrix} H_u & H_v \end{bmatrix}^T$
  - Step 3: compute gradient magnitudes (edge strength) and gradient directions (edge directions)

Original

$$H_u = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad H_v = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Magnitude

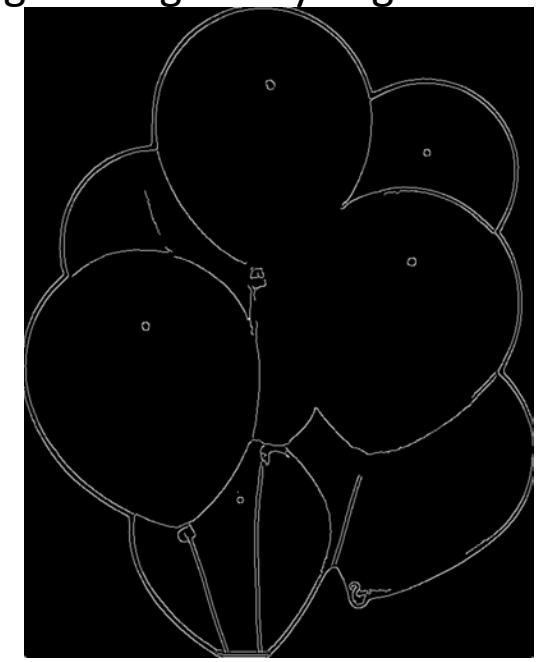
Edge Detection - Canny Edge Detector

- Canny Edge Detector
  - Step 1: Reduce noise (> Gaussian filter G_σ)
  - Step 2: Compute gradients in x and y direction (> Sobel operators H_u, H_v), gradient is a vector $\mathbf{g} = \begin{bmatrix} H_u & H_v \end{bmatrix}^T$
  - Step 3: compute gradient magnitudes (edge strength) and gradient directions (edge directions)
  - Step 4: select points with maximum magnitudes (above threshold T1)
  - Step 5: non-maximum suppression with hysteresis:
    - For all points with magnitudes above high-threshold T1
    - Trace along the edge (in edge direction) suppress all pixels that are not on traced line (pixel by pixel) and mark as visited
    - Stop tracing when point's magnitude is below low-threshold T2 or already visited

Original

$$H_u = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad H_v = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Edges using Canny Edge Detector



Magnitude

Corner Detection - Shi-Tomasi / Harris detector

- Corner Detection
  - Step 1 (Gradients): For each pixel, the algorithm calculates the image's x and y derivatives S_x, S_y (gradients) using filters like Sobel.

$$H_u = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad H_v = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Corner Detection - Shi-Tomasi / Harris detector

- Corner Detection
  - Step 1 (Gradients): For each pixel, the algorithm calculates the image's x and y derivatives S_x, S_y (gradients) using filters like Sobel.

$$H_u = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad H_v = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Corner Detection - Shi-Tomasi / Harris detector

- Corner Detection
  - Step 1 (Gradients): For each pixel, the algorithm calculates the image's x and y derivatives S_x, S_y (gradients) using filters like Sobel.
  - Step 2 (Second Moment Matrix): The derivatives are used to compute the second moment matrix (M) for a small window around the pixel:

$$H_u = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad H_v = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

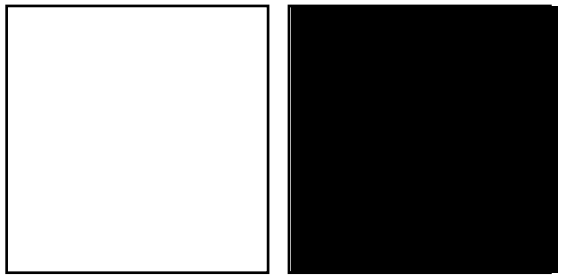
$$M = \begin{bmatrix} A & C \\ C & B \end{bmatrix} = \begin{bmatrix} (S_x)^2 & S_x S_y \\ S_x S_y & (S_y)^2 \end{bmatrix}$$

$$A = S_x^2, B = S_y^2, C = S_x S_y$$

Corner Detection - Shi-Tomasi / Harris detector

- Corner Detection
  - Step 1 (Gradients): For each pixel, the algorithm calculates the image's x and y derivatives S_x, S_y (gradients) using filters like Sobel.
  - Step 2 (Second Moment Matrix): The derivatives are used to compute the second moment matrix (M) for a small window around the pixel:
    - Properties of M:
      - Constant windows: A and B are low, C is low

$$H_u = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad H_v = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$



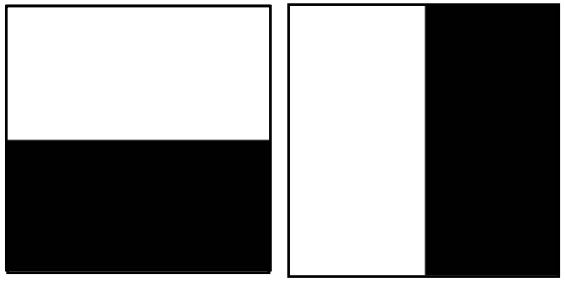
$$M = \begin{bmatrix} A & C \\ C & B \end{bmatrix} = \begin{bmatrix} (S_x)^2 & S_x S_y \\ S_x S_y & (S_y)^2 \end{bmatrix}$$

$$A = S_x^2, B = S_y^2, C = S_x S_y$$

Corner Detection - Shi-Tomasi / Harris detector

- Corner Detection
  - Step 1 (Gradients): For each pixel, the algorithm calculates the image's x and y derivatives S_x, S_y (gradients) using filters like Sobel.
  - Step 2 (Second Moment Matrix): The derivatives are used to compute the second moment matrix (M) for a small window around the pixel:
    - Properties of M:
      - Constant windows: A and B are low, C is low
      - Hori./vert. edge: A or B is high, C is low

$$H_u = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad H_v = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$



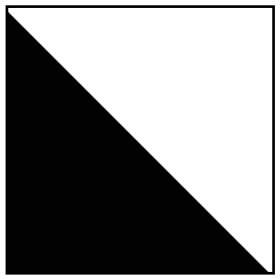
$$M = \begin{bmatrix} A & C \\ C & B \end{bmatrix} = \begin{bmatrix} (S_x)^2 & S_x S_y \\ S_x S_y & (S_y)^2 \end{bmatrix}$$

$$A = S_x^2, B = S_y^2, C = S_x S_y$$

Corner Detection - Shi-Tomasi / Harris detector

- Corner Detection
  - Step 1 (Gradients): For each pixel, the algorithm calculates the image's x and y derivatives S_x, S_y (gradients) using filters like Sobel.
  - Step 2 (Second Moment Matrix): The derivatives are used to compute the second moment matrix (M) for a small window around the pixel:
    - Properties of M:
      - Constant windows: A and B are low, C is low
      - Hori./vert. edge: A or B is high, C is low
      - Diag. edge: A,B and C are high (removed later)

$$H_u = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad H_v = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

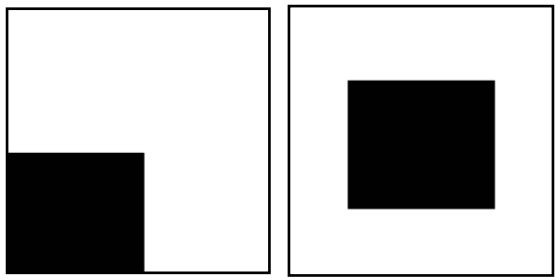


$$M = \begin{bmatrix} A & C \\ C & B \end{bmatrix} = \begin{bmatrix} (S_x)^2 & S_x S_y \\ S_x S_y & (S_y)^2 \end{bmatrix}$$

$$A = S_x^2, B = S_y^2, C = S_x S_y$$

Corner Detection - Shi-Tomasi / Harris detector

- Corner Detection
  - Step 1 (Gradients): For each pixel, the algorithm calculates the image's x and y derivatives S_x, S_y (gradients) using filters like Sobel.
  - Step 2 (Second Moment Matrix): The derivatives are used to compute the second moment matrix (M) for a small window around the pixel:
    - Properties of M:
      - Constant windows: A and B are low, C is low
      - Hori./vert. edge: A or B is high, C is low
      - Diag. edge: A,B and C are high (removed later)
      - Corner: A,B and C are high



$$H_u = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad H_v = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

$$M = \begin{bmatrix} A & C \\ C & B \end{bmatrix} = \begin{bmatrix} (S_x)^2 & S_x S_y \\ S_x S_y & (S_y)^2 \end{bmatrix}$$

$$A = S_x^2, B = S_y^2, C = S_x S_y$$

Corner Detection - Shi-Tomasi / Harris detector

- Corner Detection
  - Step 1 (Gradients): For each pixel, the algorithm calculates the image's x and y derivatives S_x, S_y (gradients) using filters like Sobel.
  - Step 2 (Second Moment Matrix): The derivatives are used to compute the second moment matrix (M) for a small window around the pixel:
    - Properties of M:
      - Constant windows: A and B are low, C is low
      - Hori./vert. edge: A or B is high, C is low
      - Diag. edge: A,B and C are high (removed later)
      - Corner: A,B and C are high
  - Step 3 (Evaluation of M): Computing of Eigenvalues (λ) for M (Shi-Tomasi) or approximation using determinant and trace of M (Harris), can be used to correctly remove diagonal edge

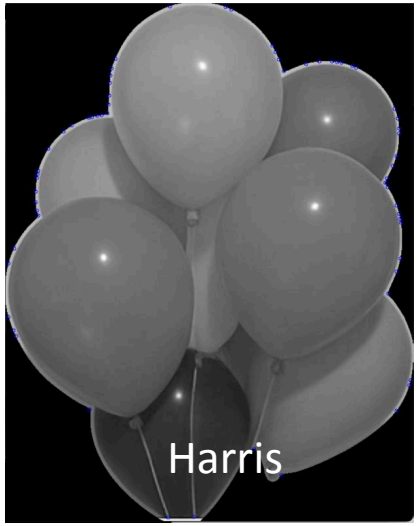
$$H_u = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad H_v = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

$$M = \begin{bmatrix} A & C \\ C & B \end{bmatrix} = \begin{bmatrix} (S_x)^2 & S_x S_y \\ S_x S_y & (S_y)^2 \end{bmatrix}$$

$$A = S_x^2, B = S_y^2, C = S_x S_y$$

Corner Detection - Shi-Tomasi / Harris detector

- Corner Detection
  - Step 1 (Gradients): For each pixel, the algorithm calculates the image's x and y derivatives S_x, S_y (gradients) using filters like Sobel.
  - Step 2 (Second Moment Matrix): The derivatives are used to compute the second moment matrix (M) for a small window around the pixel:
    - Properties of M:
      - Constant windows: A and B are low, C is low
      - Hori./vert. edge: A or B is high, C is low
      - Diag. edge: A,B and C are high (removed later)
      - Corner: A,B and C are high
  - Step 3 (Evaluation of M): Computing of Eigenvalues (λ) for M (Shi-Tomasi) or approximation using determinant and trace of M (Harris), can be used to correctly remove diagonal edge
  - Step 4 (Corner Detection): A point is identified if the eigenvalues or their approximation are large (meaning the local image patch changes significantly in all directions)
  - 6. Post-processing:
    Non-maximal suppression to keep only the strongest corners



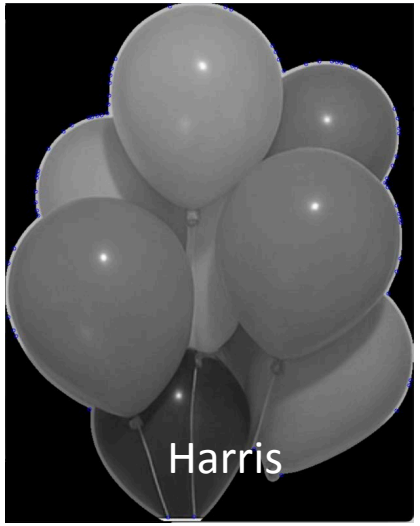
$$H_u = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad H_v = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

$$M = \begin{bmatrix} A & C \\ C & B \end{bmatrix} = \begin{bmatrix} (S_x)^2 & S_x S_y \\ S_x S_y & (S_y)^2 \end{bmatrix}$$

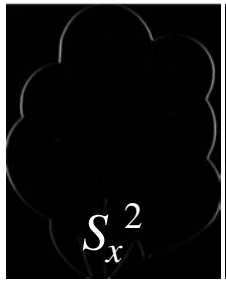
$$A = S_x^2, B = S_y^2, C = S_x S_y$$

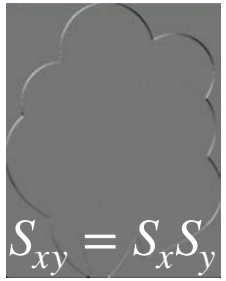
Corner Detection - Shi-Tomasi / Harris detector

- Corner Detection
  - Step 1 (Gradients): For each pixel, the algorithm calculates the image's x and y derivatives S_x, S_y (gradients) using filters like Sobel.
  - Step 2 (Second Moment Matrix): The derivatives are used to compute the second moment matrix (M) for a small window around the pixel:
    - Properties of M:
      - Constant windows: A and B are low, C is low
      - Hori./vert. edge: A or B is high, C is low
      - Diag. edge: A,B and C are high (removed later)
      - Corner: A,B and C are high
  - Step 3 (Evaluation of M): Computing of Eigenvalues (λ) for M (Shi-Tomasi) or approximation using determinant and trace of M (Harris), can be used to correctly remove diagonal edge
  - Step 4 (Corner Detection): A point is identified if the eigenvalues or their approximation are large (meaning the local image patch changes significantly in all directions)
  - 6. Post-processing:
    Non-maximal suppression to keep only the strongest corners



$$H_u = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad H_v = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$





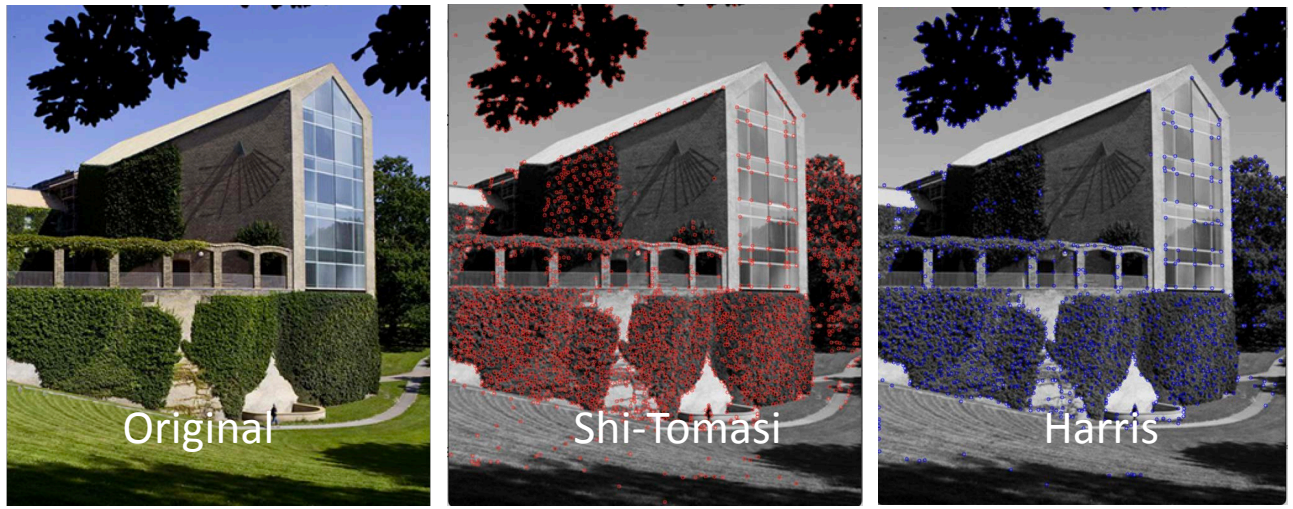
$$M = \begin{bmatrix} A & C \\ C & B \end{bmatrix} = \begin{bmatrix} (S_x)^2 & S_x S_y \\ S_x S_y & (S_y)^2 \end{bmatrix}$$

$$A = S_x^2, B = S_y^2, C = S_x S_y$$

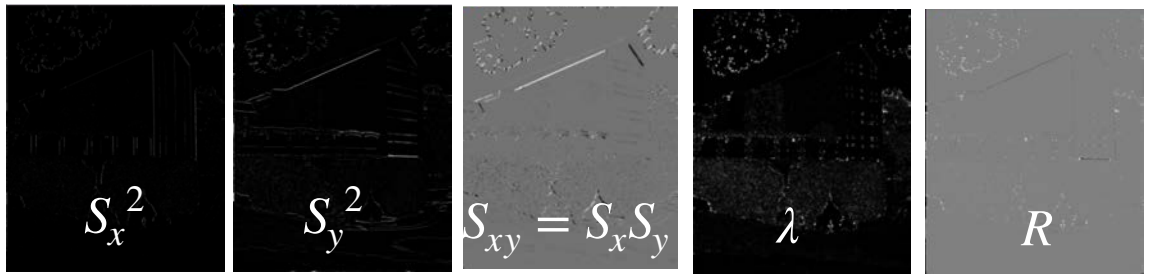
Corner Detection - Shi-Tomasi / Harris detector

Corner Detection - Shi-Tomasi / Harris detector

- Corner Detection
  - Step 1 (Gradients): For each pixel, the algorithm calculates the image's x and y derivatives S_x, S_y (gradients) using filters like Sobel.
  - Step 2 (Second Moment Matrix): The derivatives are used to compute the second moment matrix (M) for a small window around the pixel:
    - Properties of M:
      - Constant windows: A and B are low, C is low
      - Hori./vert. edge: A or B is high, C is low
      - Diag. edge: A,B and C are high (removed later)
      - Corner: A,B and C are high
  - Step 3 (Evaluation of M): Computing of Eigenvalues (λ) for M (Shi-Tomasi) or approximation using determinant and trace of M (Harris), can be used to correctly remove diagonal edge
  - Step 4 (Corner Detection): A point is identified if the eigenvalues or their approximation are large (meaning the local image patch changes significantly in all directions)
  - 6. Post-processing:
    Non-maximal suppression to keep only the strongest corners



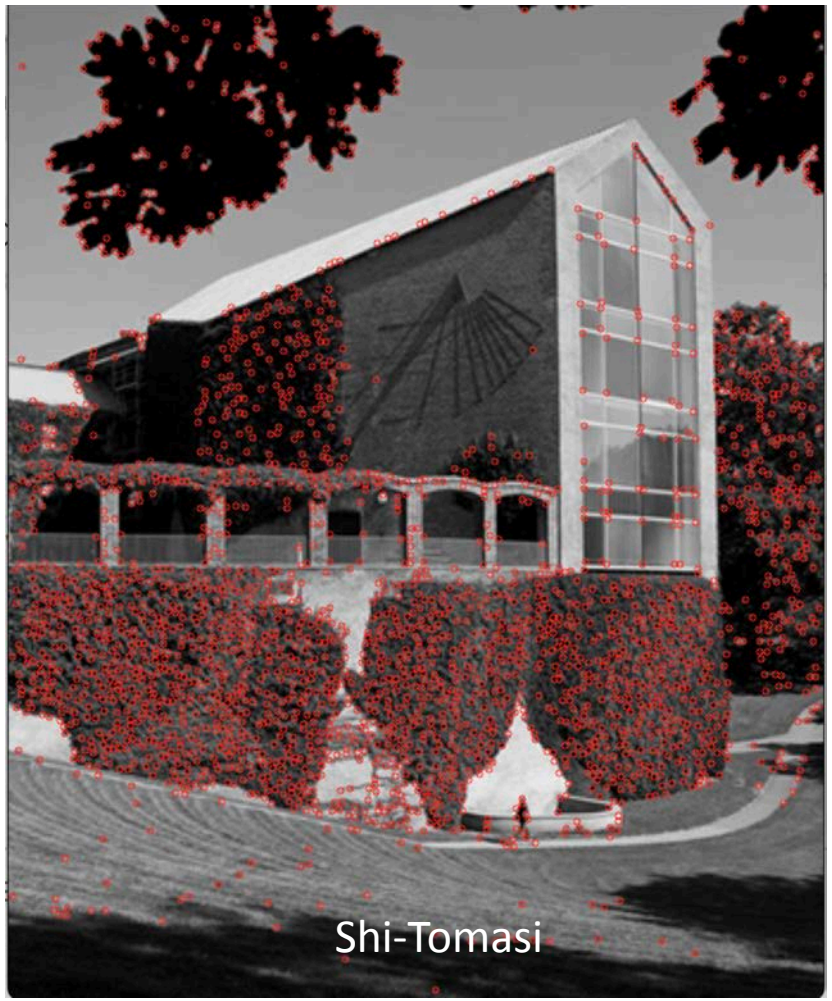
$$H_u = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad H_v = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$



$$M = \begin{bmatrix} A & C \\ C & B \end{bmatrix} = \begin{bmatrix} (S_x)^2 & S_x S_y \\ S_x S_y & (S_y)^2 \end{bmatrix}$$

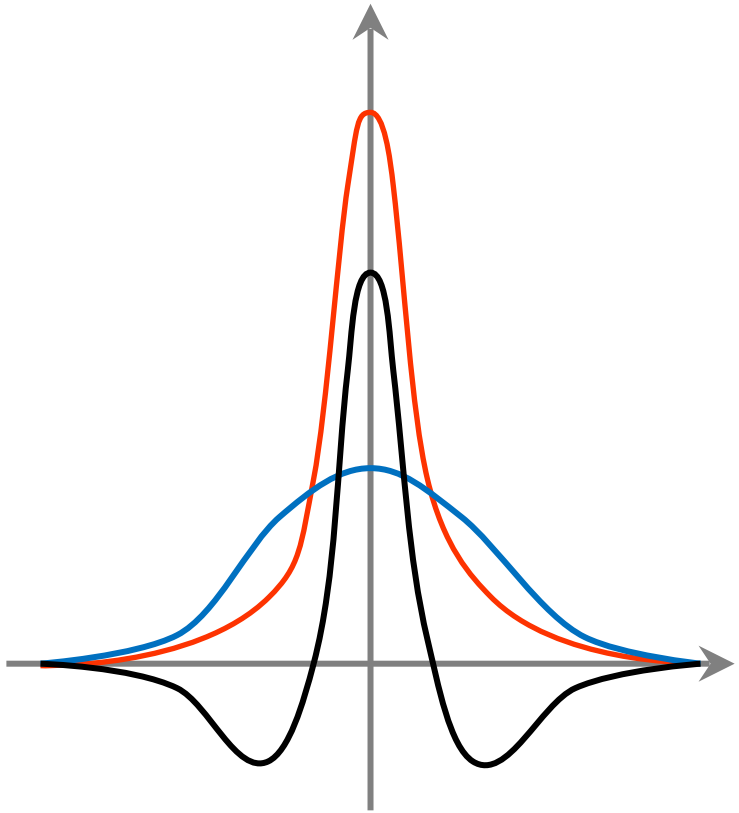
$$A = S_x^2, B = S_y^2, C = S_x S_y$$

Corner Detection - Shi-Tomasi / Harris detector



Blob Detection - Difference of Gaussians (DoG)

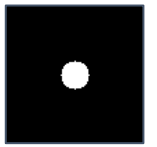
- ‘Blobs’ are alternative features
  - A blob is a dark region surrounded by a bright region, or vice-versa
  - Blobs have a scale as well as a location
- Difference of Gaussian filter
  - Two Gaussians of different width
  - Subtract wide from narrow
  - Bright blobs – high positive response
  - Dark blobs – high negative response



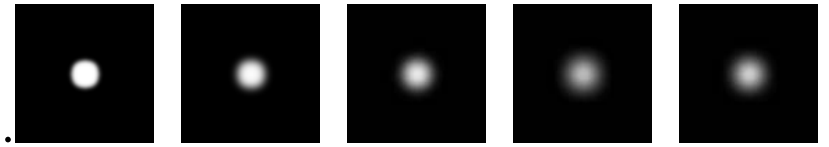
Blob Detection - Difference of Gaussians (DoG)

- ‘Blobs’ are alternative features
  - A blob is a dark region surrounded by a bright region, or vice-versa
  - Blobs have a scale as well as a location
- Difference of Gaussian filter
  - Two Gaussians of different width
  - Subtract wide from narrow
  - Bright blobs – high positive response
  - Dark blobs – high negative response

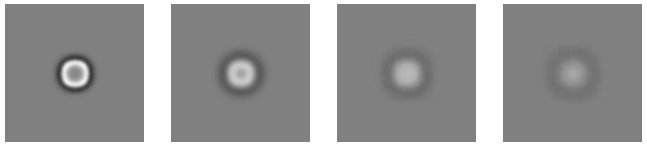
Original Image



Gaussian blur with $\sigma = 1.5,\ 3, 4.5,\ 6, ..$

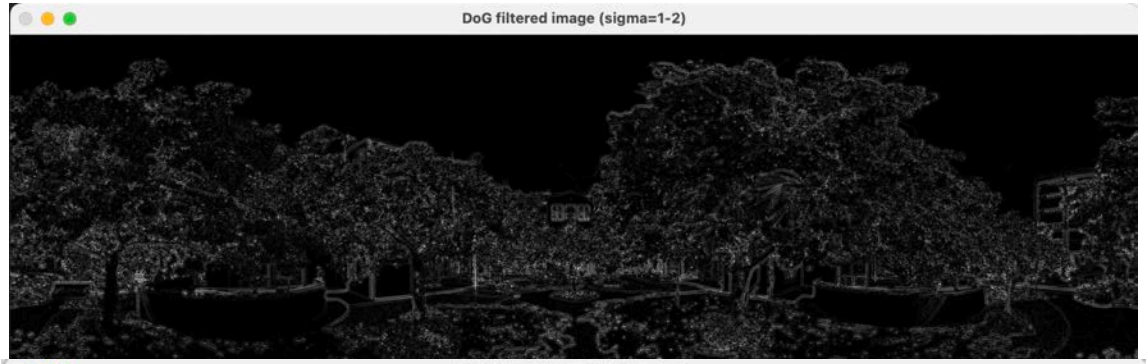


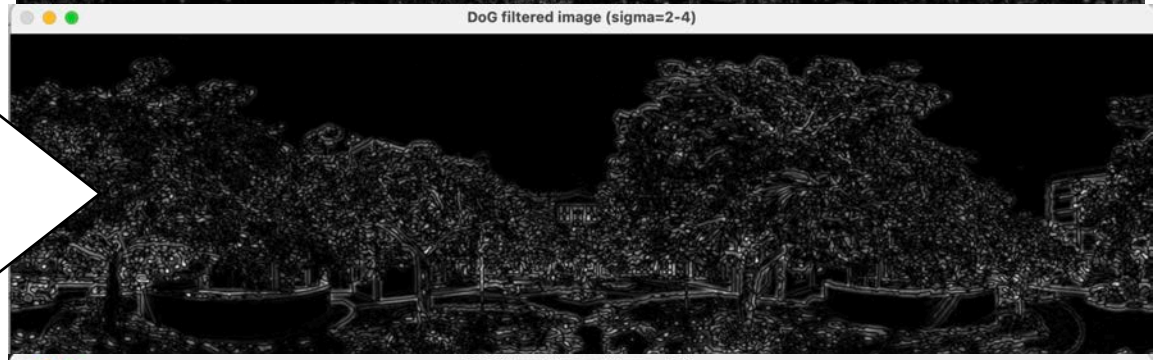
Difference of Gaussians



Blob Detection

DoG



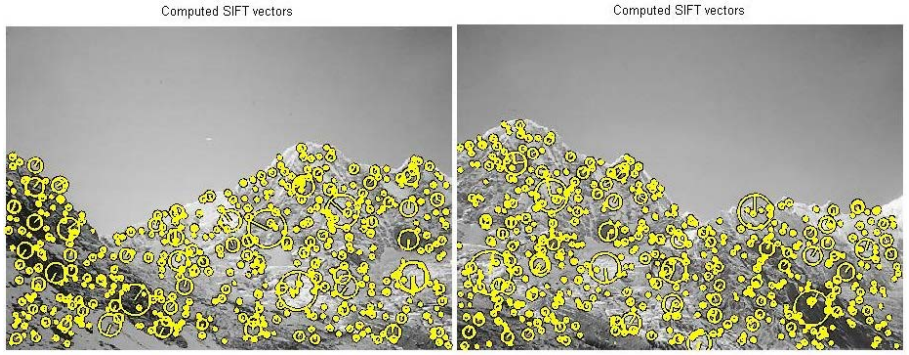


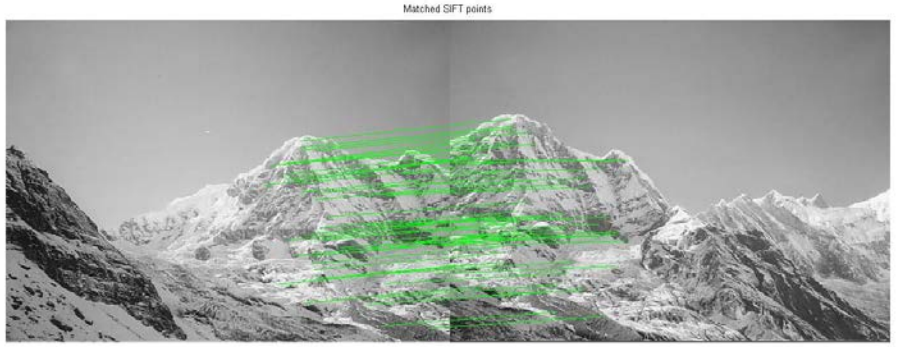
Lets put it all together -
Scale Invariant Features Transform (SIFT)

Scale Invariant Features Transform (SIFT)

- Common problems:
  - Panorama stitching
  - Image Search
  - Object recognition
  - Tracking (covered a different time)

- Scale Invariant Features Transform (SIFT):
  - Computer vision algorithm to
    - Detect,
    - Describe,
    - And match local features in images





Scale Invariant Features Transform (SIFT)

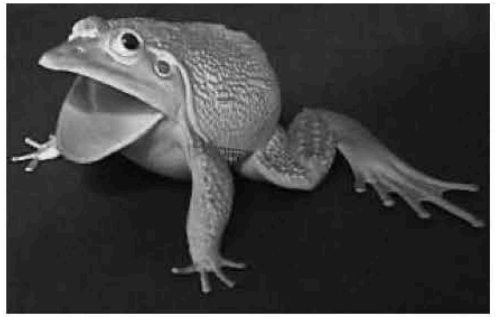
- Common problems:
  - Panorama stitching
  - Image Search
  - Object recognition
  - Tracking (covered a different time)

- Scale Invariant Features Transform (SIFT):
  - Computer vision algorithm to
    - Detect,
    - Describe,
    - And match local features in images

Scale Invariant Features Transform (SIFT)

- Common problems:
  - Panorama stitching
  - Image Search
  - Object recognition
  - Tracking (covered a different time)

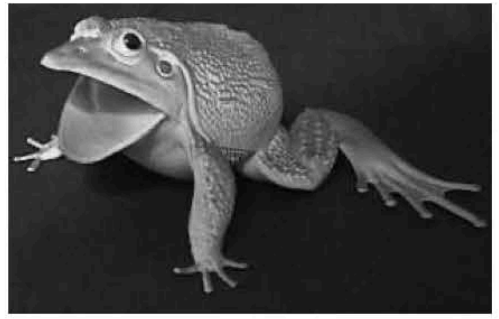
- Scale Invariant Features Transform (SIFT):
  - Computer vision algorithm to
    - Detect,
    - Describe,
    - And match local features in images



Scale Invariant Features Transform (SIFT)

- Common problems:
  - Panorama stitching
  - Image Search
  - Object recognition
  - Tracking (covered a different time)

- Scale Invariant Features Transform (SIFT):
  - Computer vision algorithm to
    - Detect,
    - Describe,
    - And match local features in images



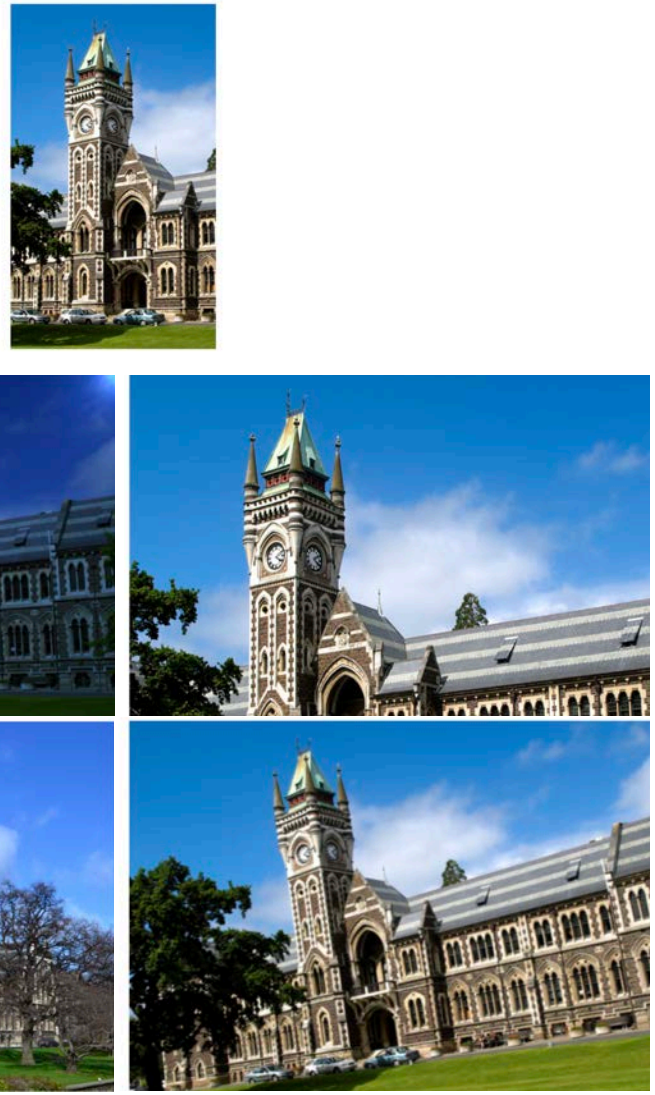
Scale Invariant Features Transform (SIFT)

- Common problems:
  - Panorama stitching
  - Image Search
  - Object recognition
  - Tracking (covered a different time)

- Scale Invariant Features Transform (SIFT):
  - Computer vision algorithm to
    - Detect,
    - Describe,
    - And match local features in images

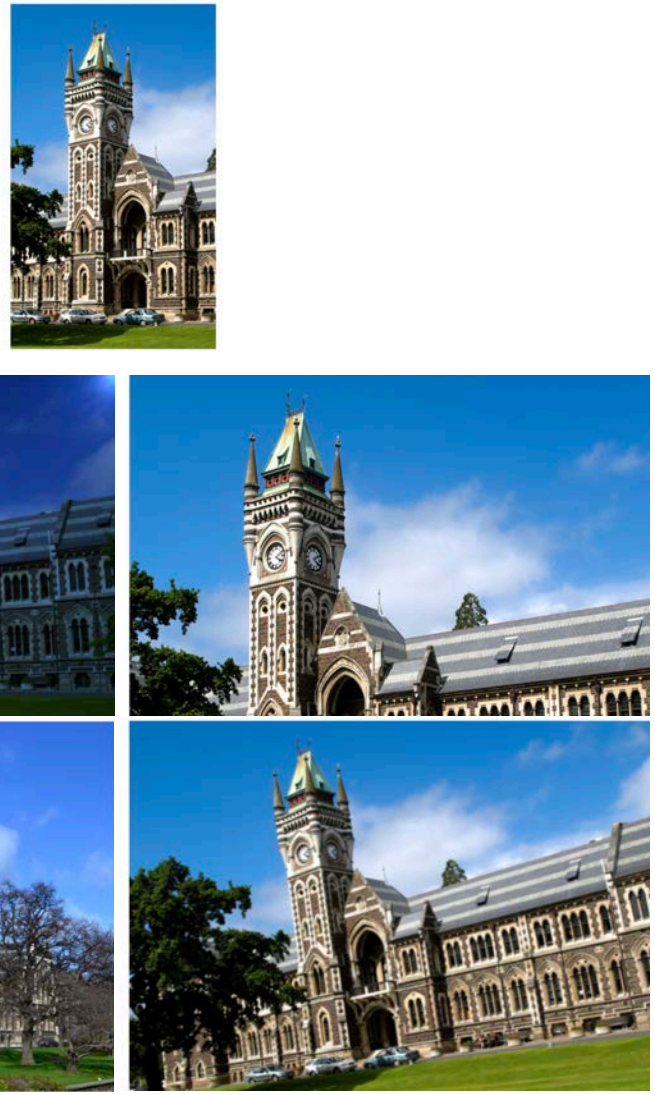
Scale Invariant Features Transform (SIFT)

- Properties of SIFT
  - **Robustness:** Invariance to changes in illumination, scale, rotation, affine, perspective.
  - **Locality:** robustness to occlusion and clutter.
  - **Distinctiveness:** Easy to match to a large database of objects (e.g. different images).
  - **Quantity:** Many features (points) can be generated for even small objects.
  - **Efficiency:** Computationally “cheap”, real-time performance.



Scale Invariant Features Transform (SIFT)

- Properties of SIFT
  - **Robustness:** Invariance to changes in illumination, scale, rotation, affine, perspective.
  - **Locality:** robustness to occlusion and clutter.
  - **Distinctiveness:** Easy to match to a large database of objects (e.g. different images).
  - **Quantity:** Many features (points) can be generated for even small objects.
  - **Efficiency:** Computationally “cheap”, real-time performance.



SIFT - Feature detection (blobs)

Scale Invariant Features Transform (SIFT)

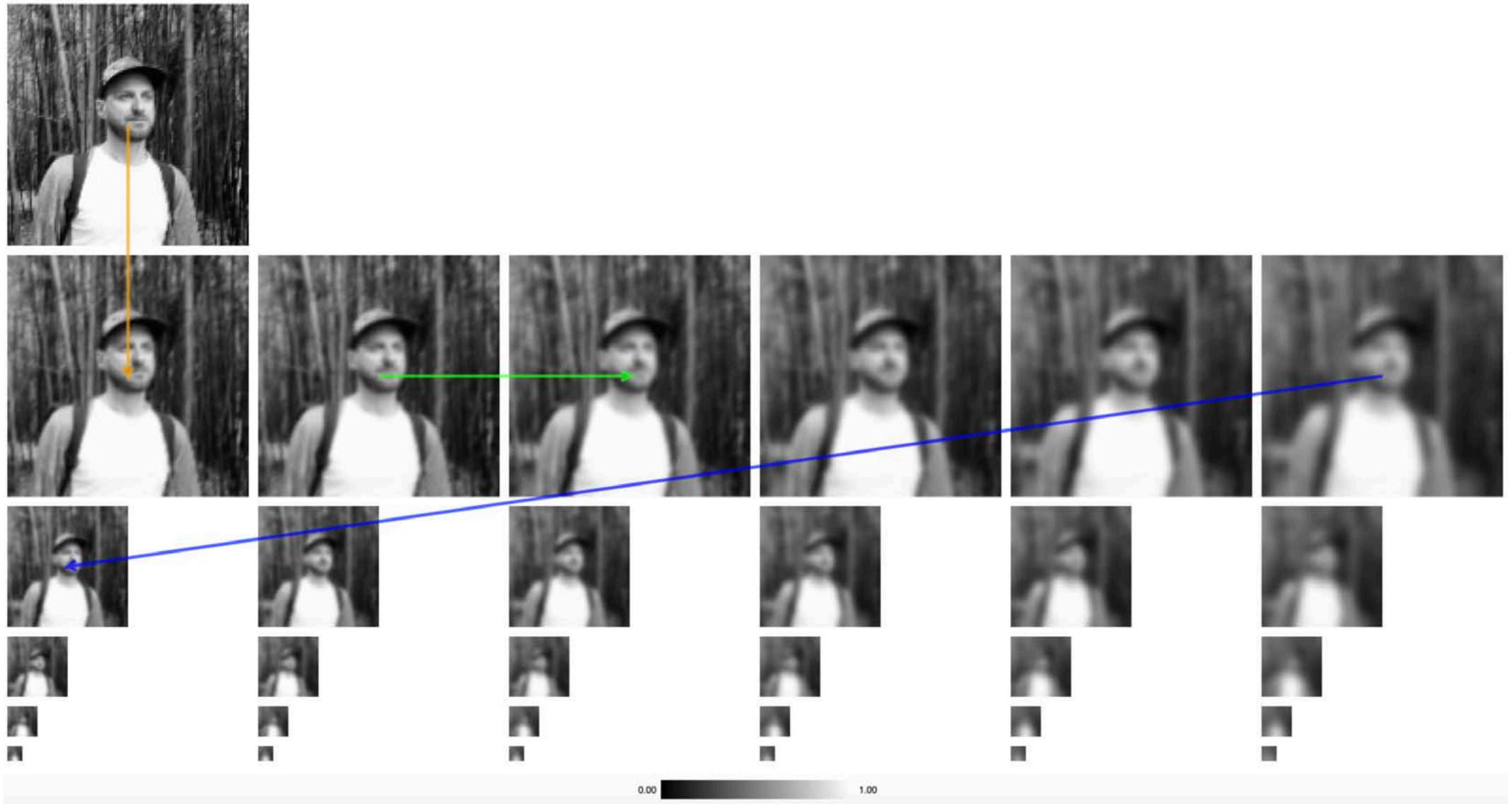
- Properties of SIFT
  - **Robustness:** Invariance to changes in illumination, scale, rotation, affine, perspective.
  - **Locality:** robustness to occlusion and clutter.
  - **Distinctiveness:** Easy to match to a large database of objects (e.g. different images).
  - **Quantity:** Many features (points) can be generated for even small objects.
  - **Efficiency:** Computationally “cheap”, real-time performance.

SIFT - Scalespace creation

Input image
128x128

Normalised

Doubled input size
using
bilinear_interpolation



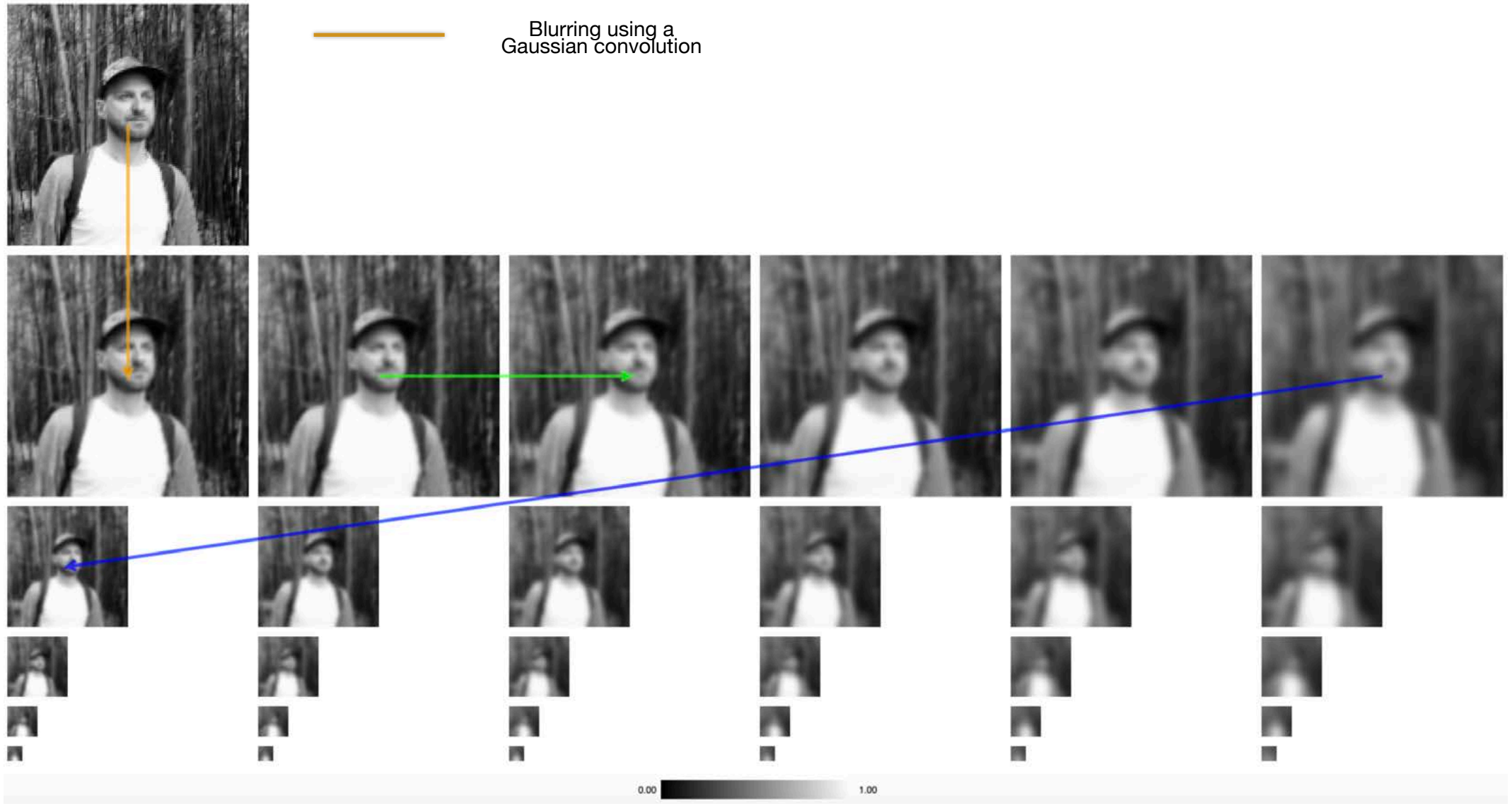
Uses some material from : https://weitz.de/sift/

SIFT - Scalespace creation

Input image
128x128

Normalised

Doubled input size
using
bilinear_interpolation



Uses some material from : https://weitz.de/sift/

SIFT - Scalespace creation

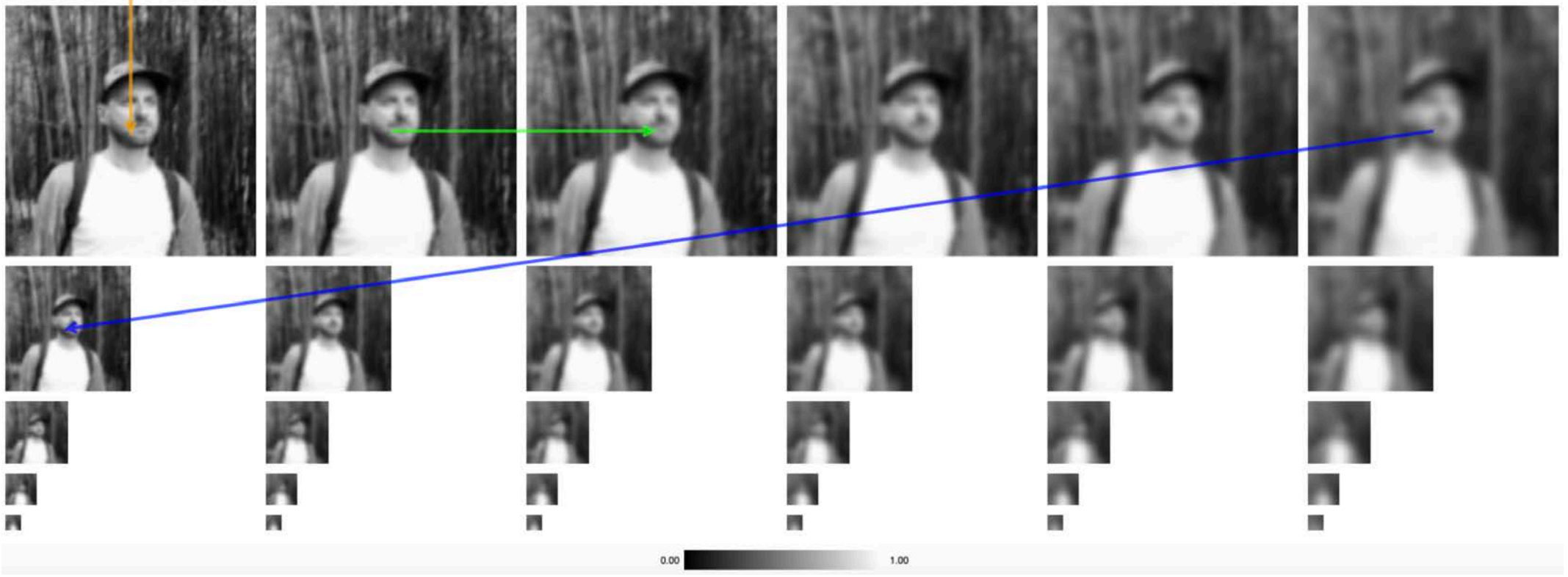
Input image
128x128

Normalised

Doubled input size
using
bilinear_interpolation

Blurring using a
Gaussian convolution

Blurring using a
Gaussian convolution
with increasing
standard deviation
(more blur)



Uses some material from : https://weitz.de/sift/

SIFT - Scalespace creation

Input image
128x128

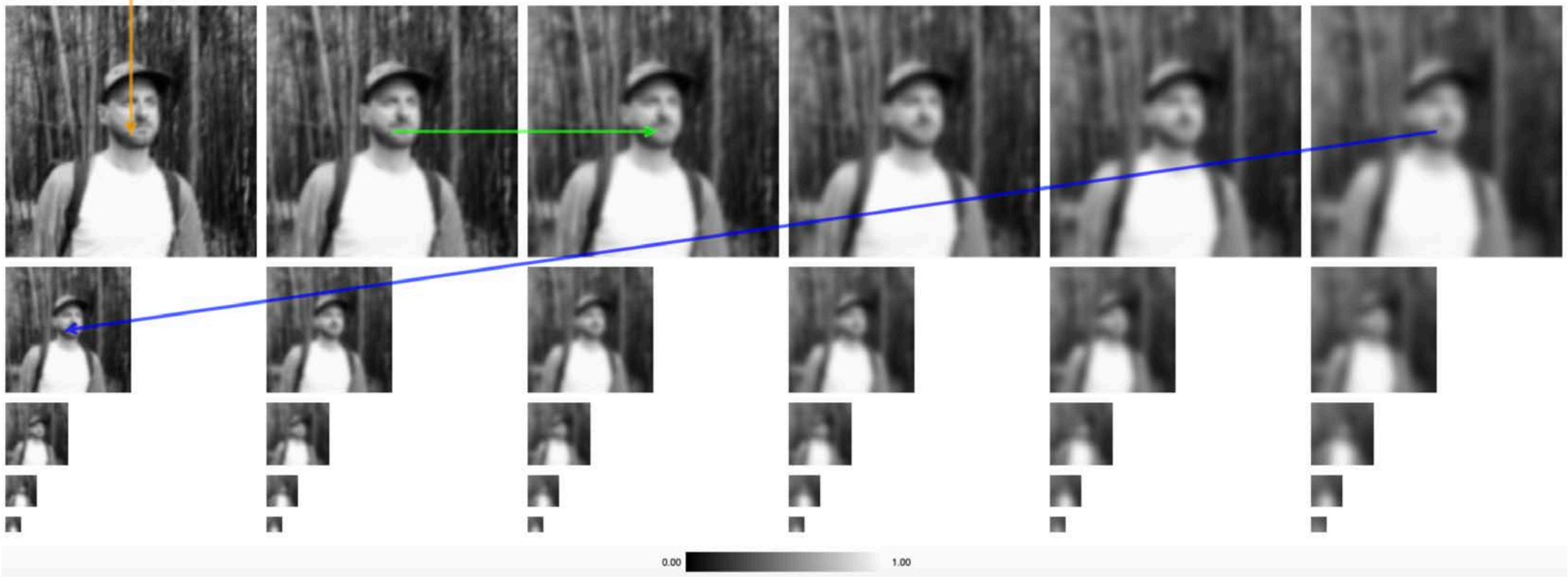
Normalised

Doubled input size
using
bilinear_interpolation

Blurring using a
Gaussian convolution

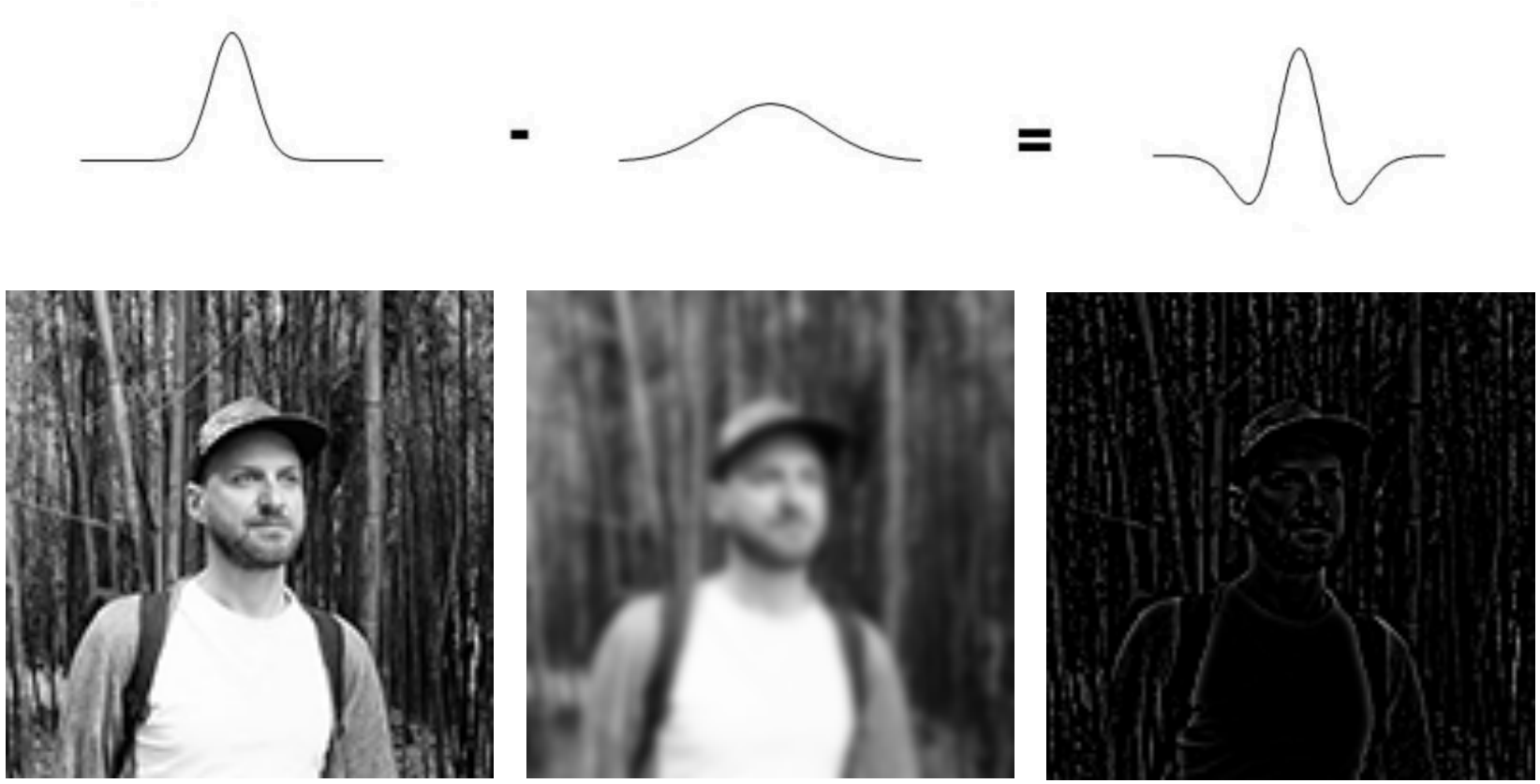
Blurring using a
Gaussian convolution
with increasing
standard deviation
(more blur)

Downsampling (size
reduction)



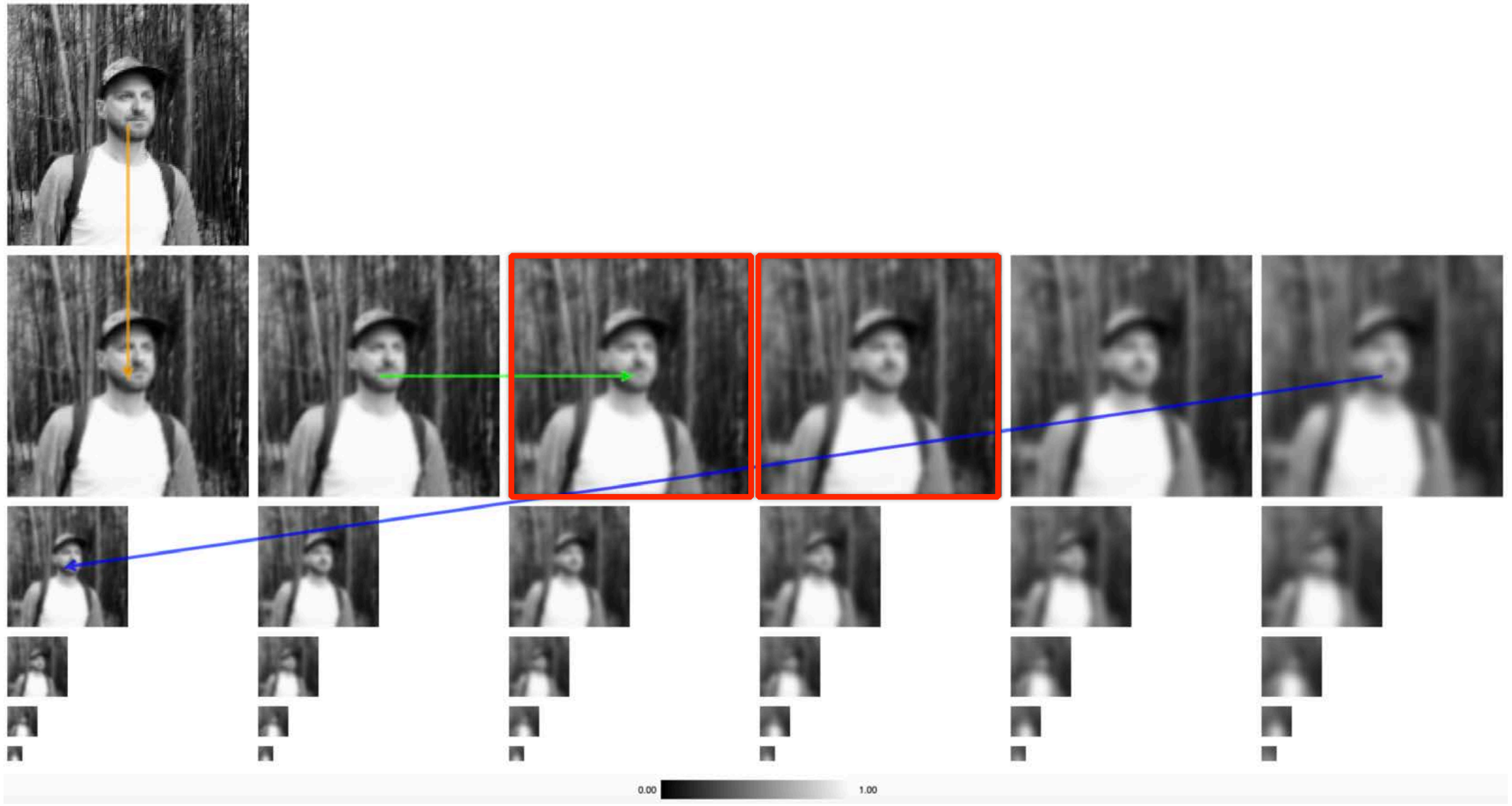
Uses some material from : https://weitz.de/sift/

SIFT - Difference of Gaussians



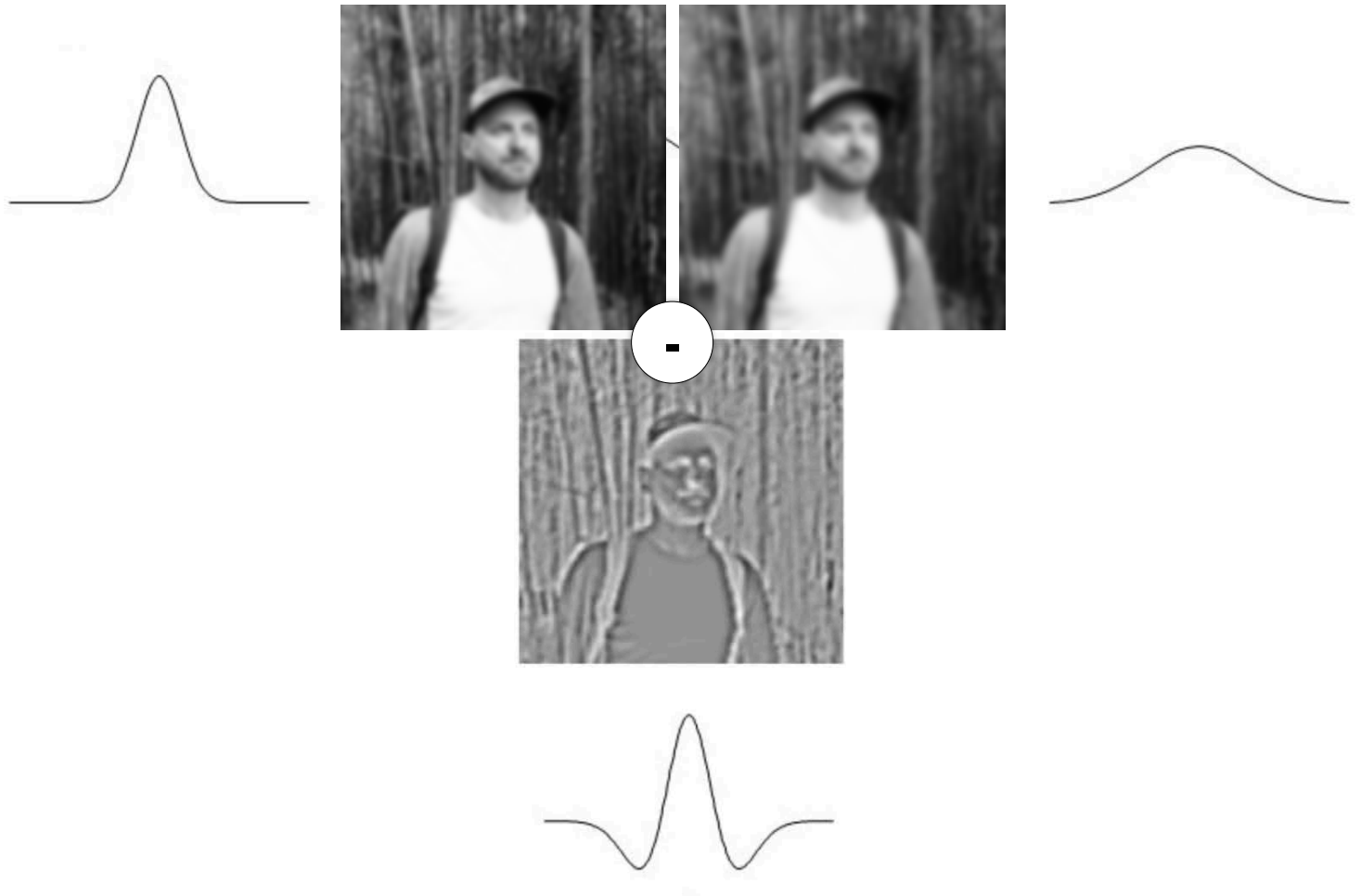
Otero, I. R. (2015). *Anatomy of the SIFT Method* (Doctoral dissertation, École normale supérieure de Cachan-ENS Cachan).

SIFT - Difference of Gaussians



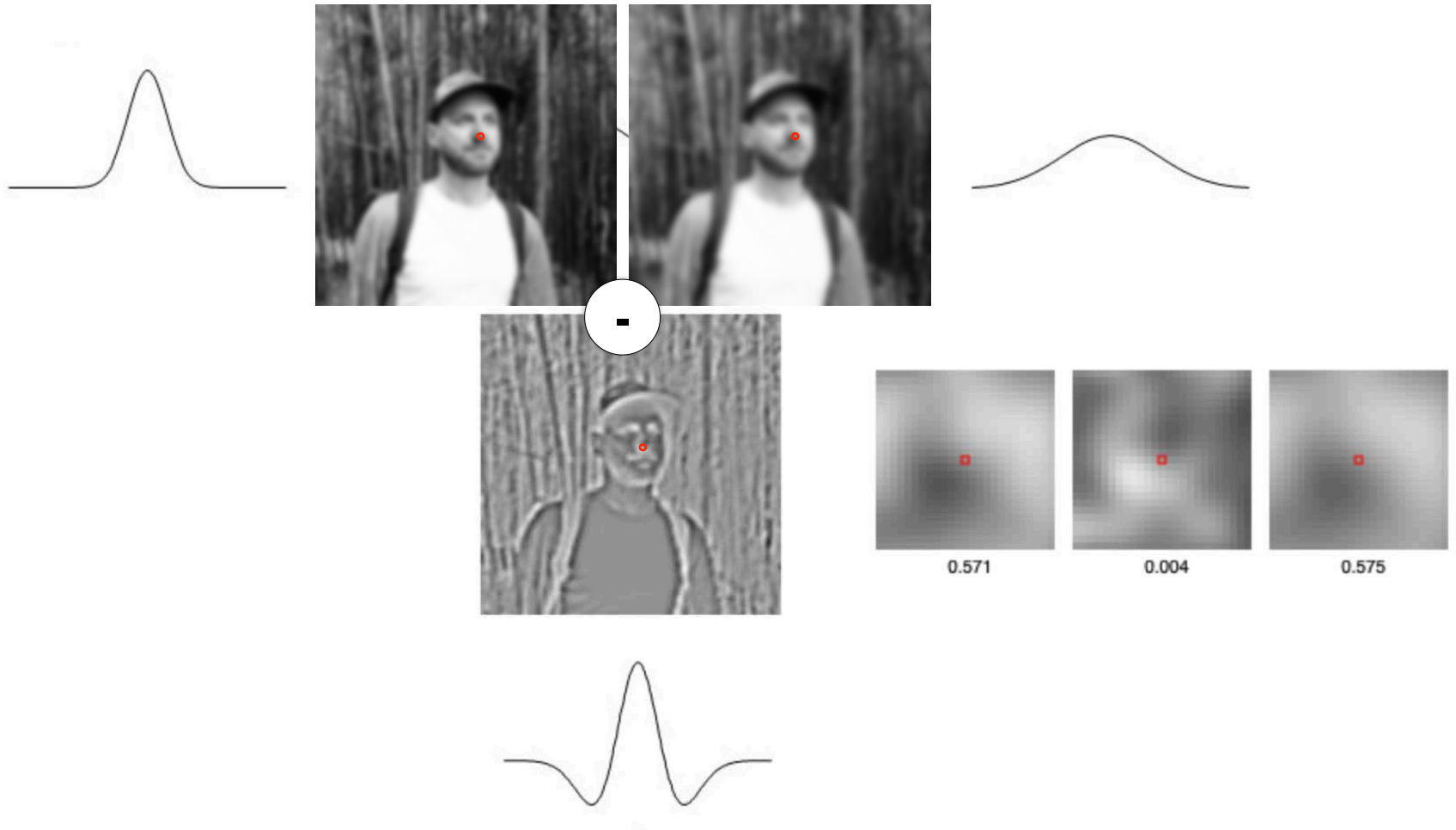
Otero, I. R. (2015). *Anatomy of the SIFT Method* (Doctoral dissertation, École normale supérieure de Cachan-ENS Cachan).

SIFT - Difference of Gaussians



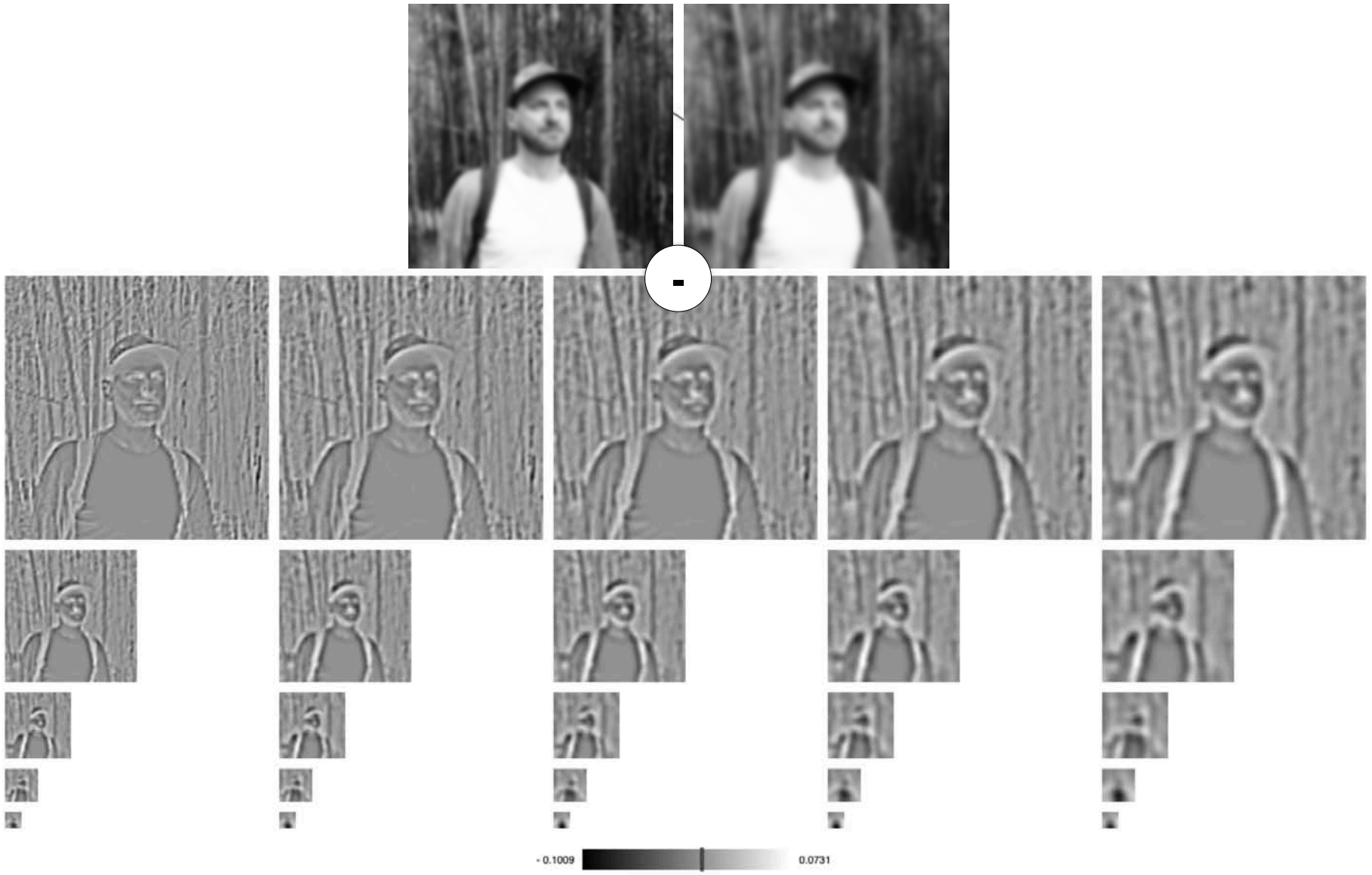
Otero, I. R. (2015). *Anatomy of the SIFT Method* (Doctoral dissertation, École normale supérieure de Cachan-ENS Cachan).

SIFT - Difference of Gaussians



Otero, I. R. (2015). *Anatomy of the SIFT Method* (Doctoral dissertation, École normale supérieure de Cachan-ENS Cachan).

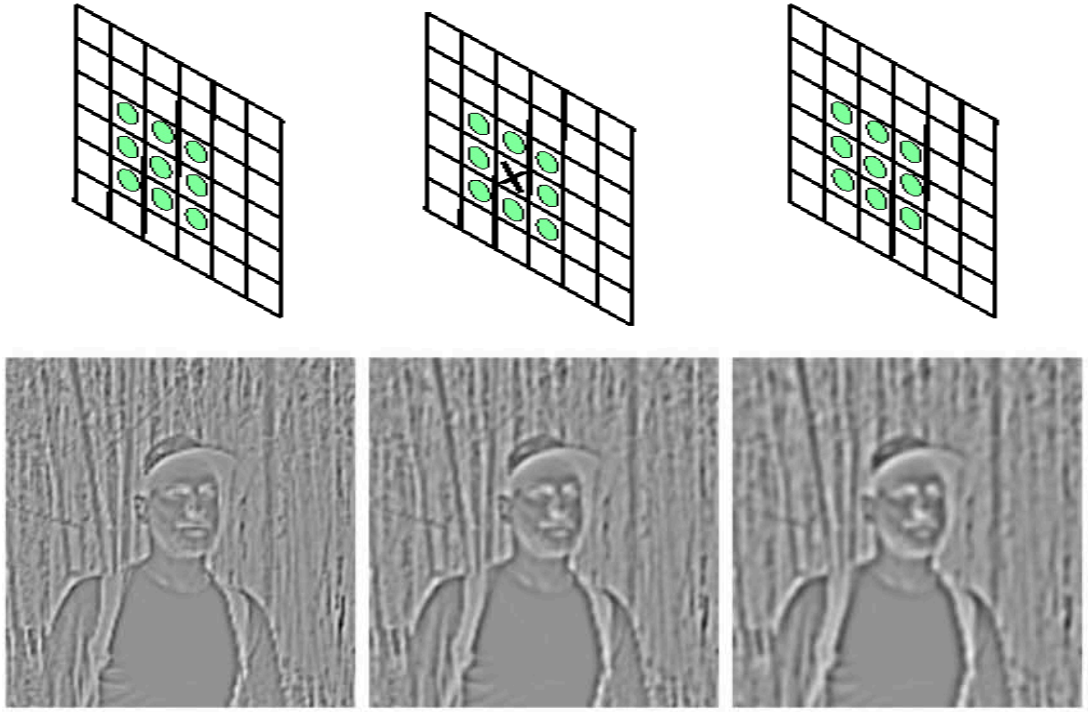
SIFT - Difference of Gaussians



Otero, I. R. (2015). *Anatomy of the SIFT Method* (Doctoral dissertation, École normale supérieure de Cachan-ENS Cachan).

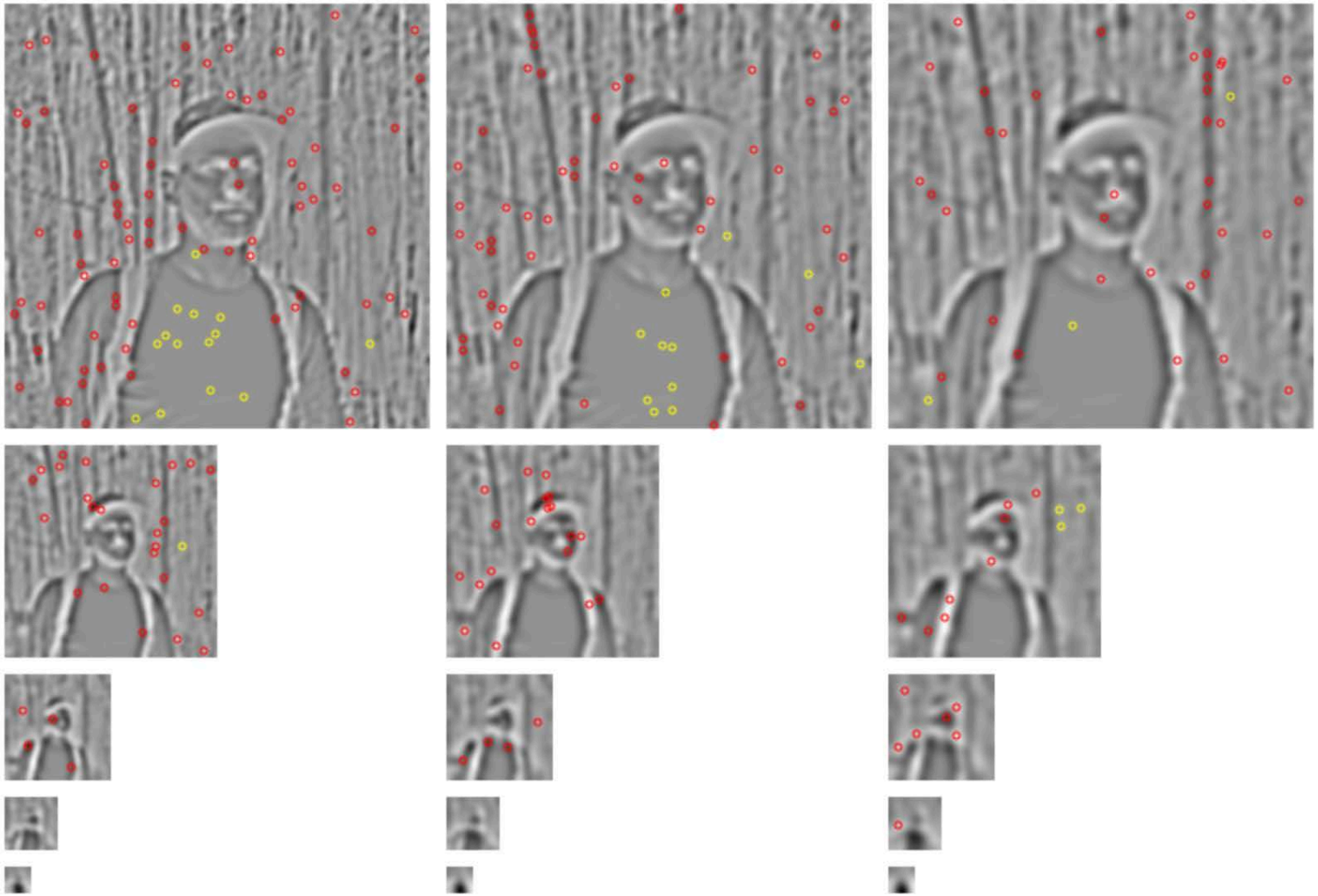
SIFT - Discrete extrema

- Pixel whose gray value is larger than those of all of its 26 neighbor pixel (Discrete Maximum)
- Pixel whose gray value is smaller than those of all of its 26 neighbor pixel (Discrete minimum)



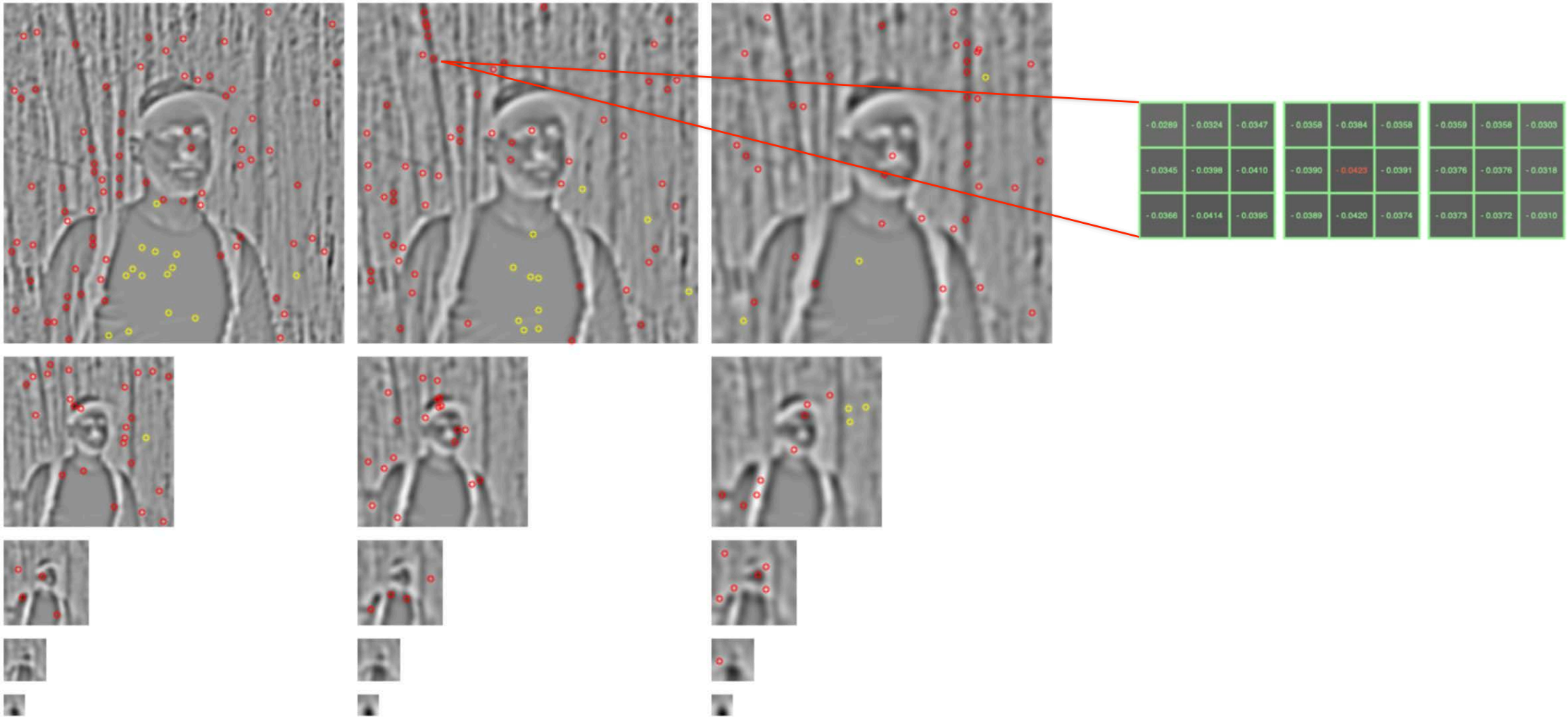
Otero, I. R. (2015). *Anatomy of the SIFT Method* (Doctoral dissertation, École normale supérieure de Cachan-ENS Cachan).

SIFT - Discrete extrema



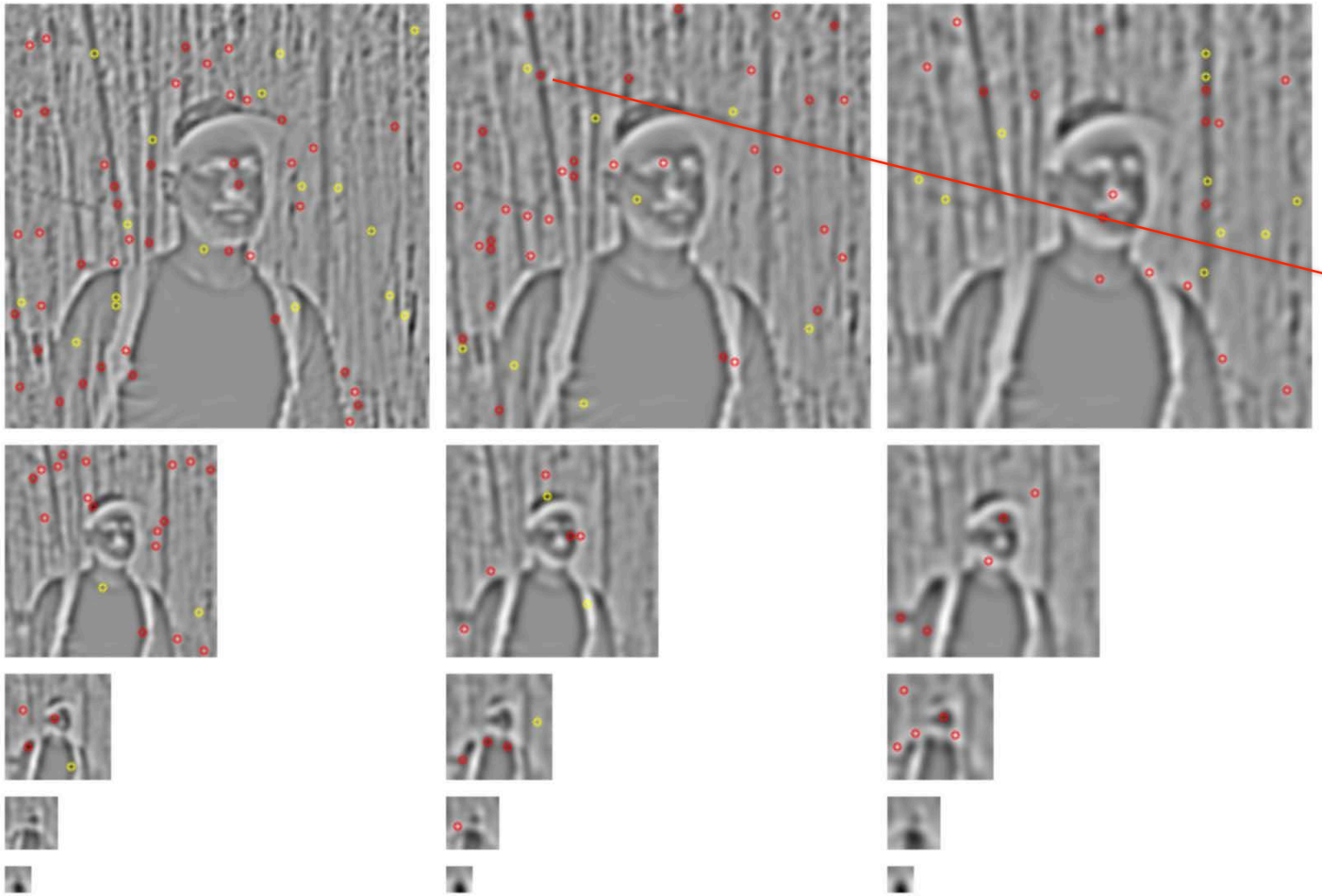
Otero, I. R. (2015). *Anatomy of the SIFT Method* (Doctoral dissertation, École normale supérieure de Cachan-ENS Cachan).

SIFT - Discrete extrema



Otero, I. R. (2015). *Anatomy of the SIFT Method* (Doctoral dissertation, École normale supérieure de Cachan-ENS Cachan).

SIFT - Discrete extrema



	discrete	interpolated
x	28.5	28.571
y	21.0	21.278
scale	2	1.641

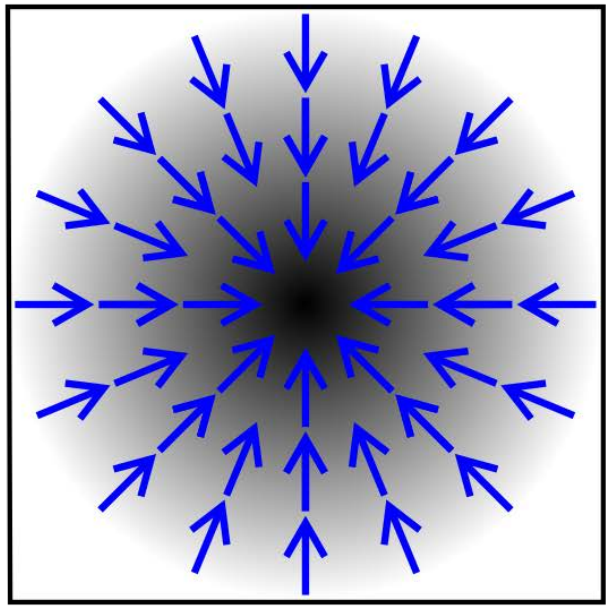
- We discard points (yellow) with small absolute values (noise) or extremas on an edge

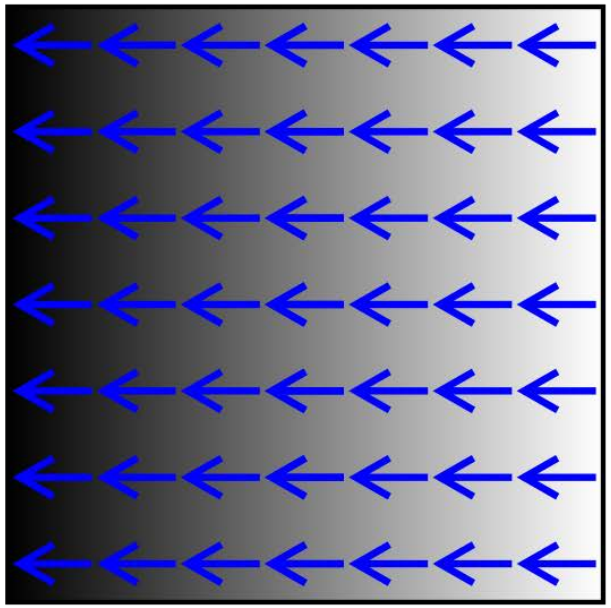
Otero, I. R. (2015). *Anatomy of the SIFT Method* (Doctoral dissertation, École normale supérieure de Cachan-ENS Cachan).

SIFT - Feature description

SIFT - Feature description

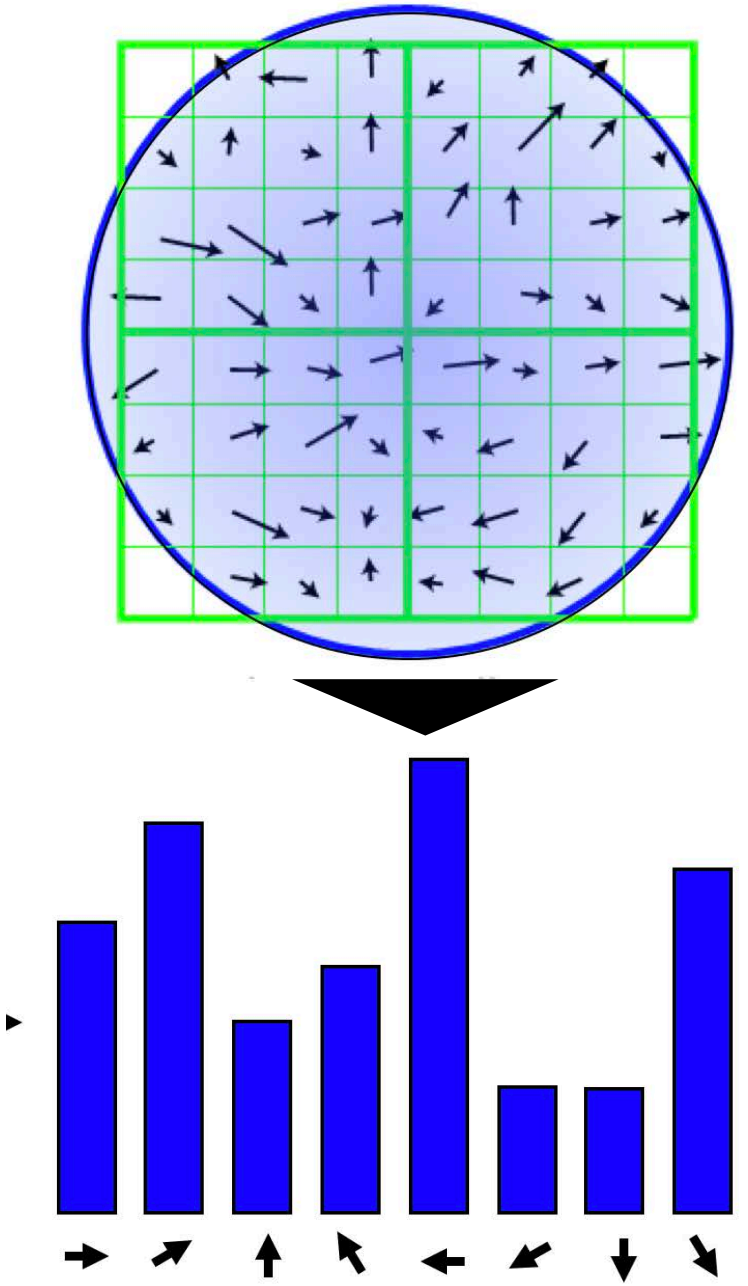
- Feature description (orientation and gradient)
- For each feature (corner) we compute gradient's magnitude and orientation
- Image gradients:





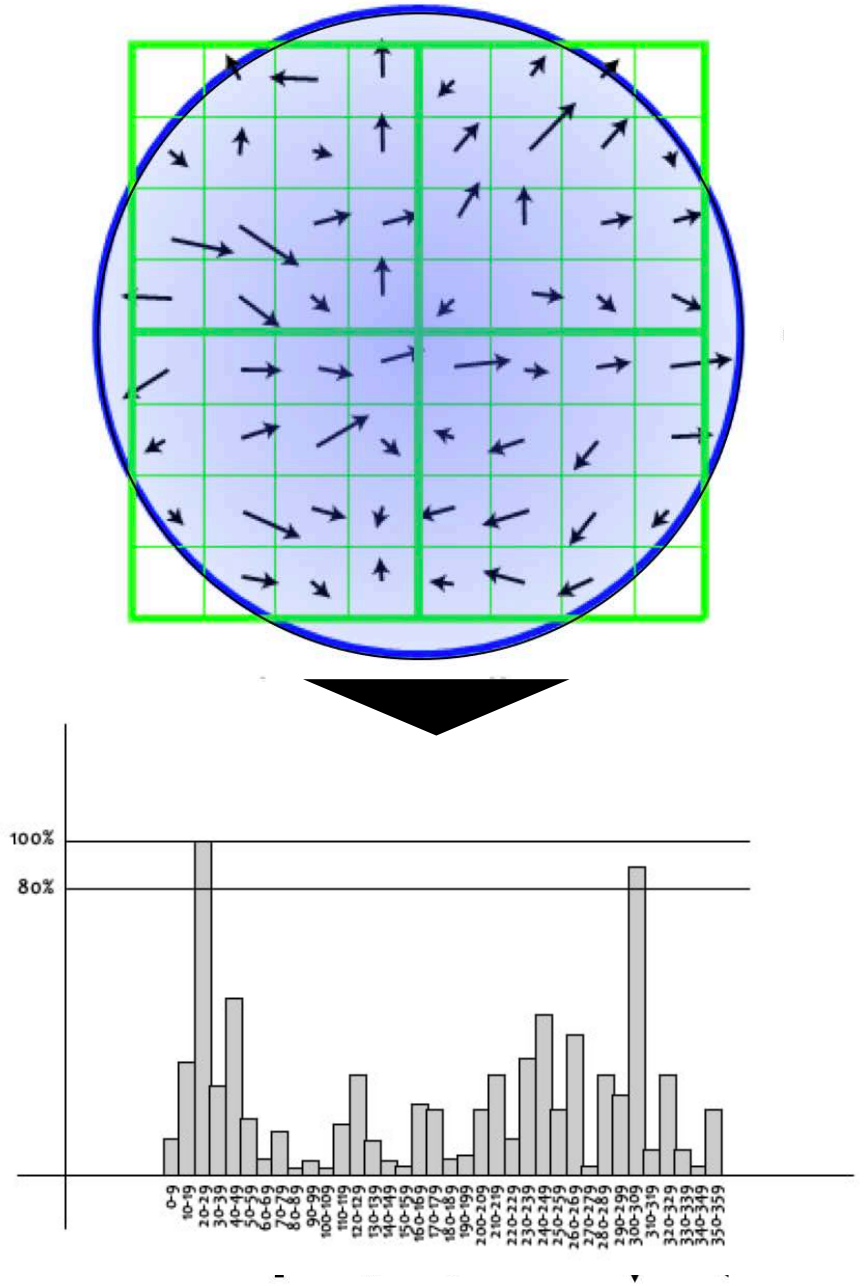
SIFT - Feature description

- Feature description (orientation and gradient)
- Create gradient histogram (8 bins, simplified) weighted by magnitude and Gaussian window
- Any histogram peak within 80% of highest peak is assigned to keypoint (multiple assignments possible)
- Result is keypoints/features main orientation

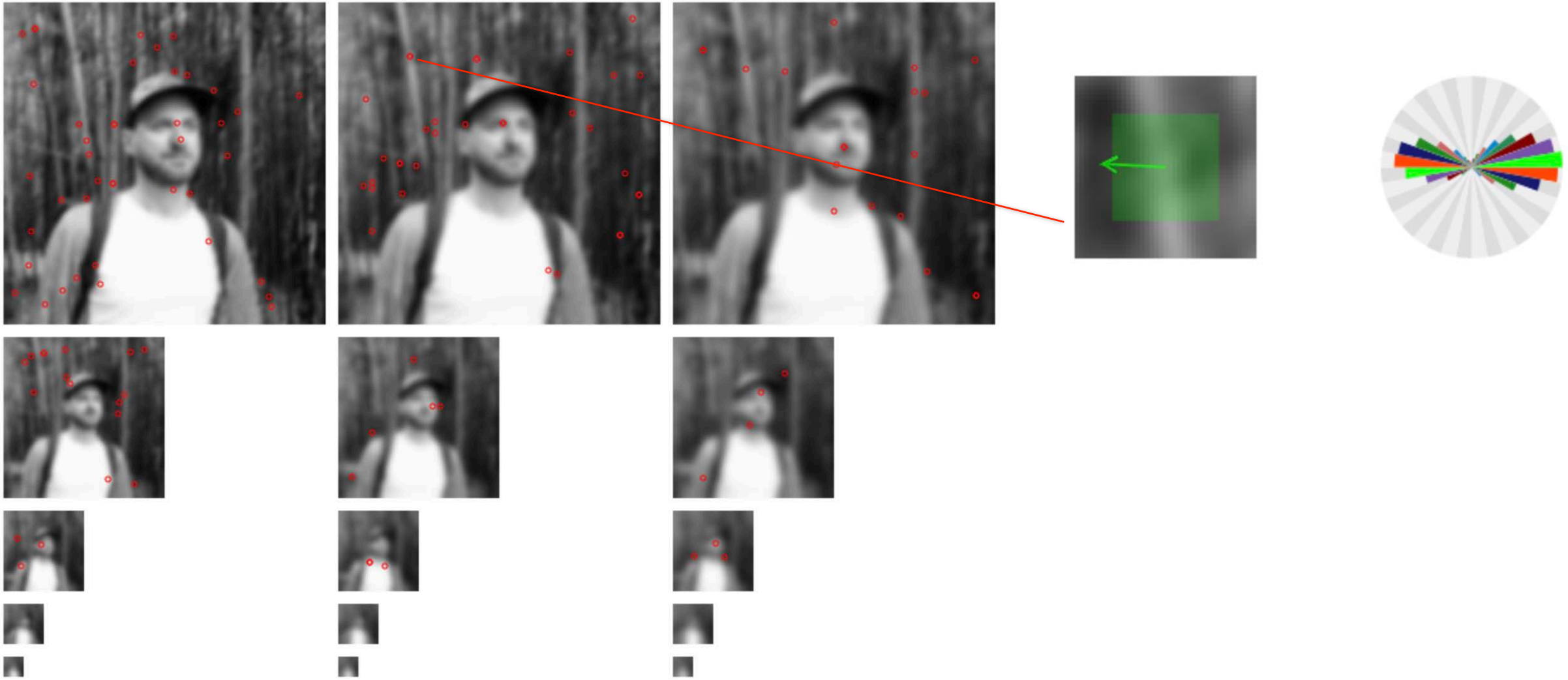


SIFT - Feature description

- Feature description (orientation and gradient)
- Create gradient histogram (8 bins, simplified) weighted by magnitude and Gaussian window
- Any histogram peak within 80% of highest peak is assigned to keypoint (multiple assignments possible)
- Result is keypoints/features main orientation

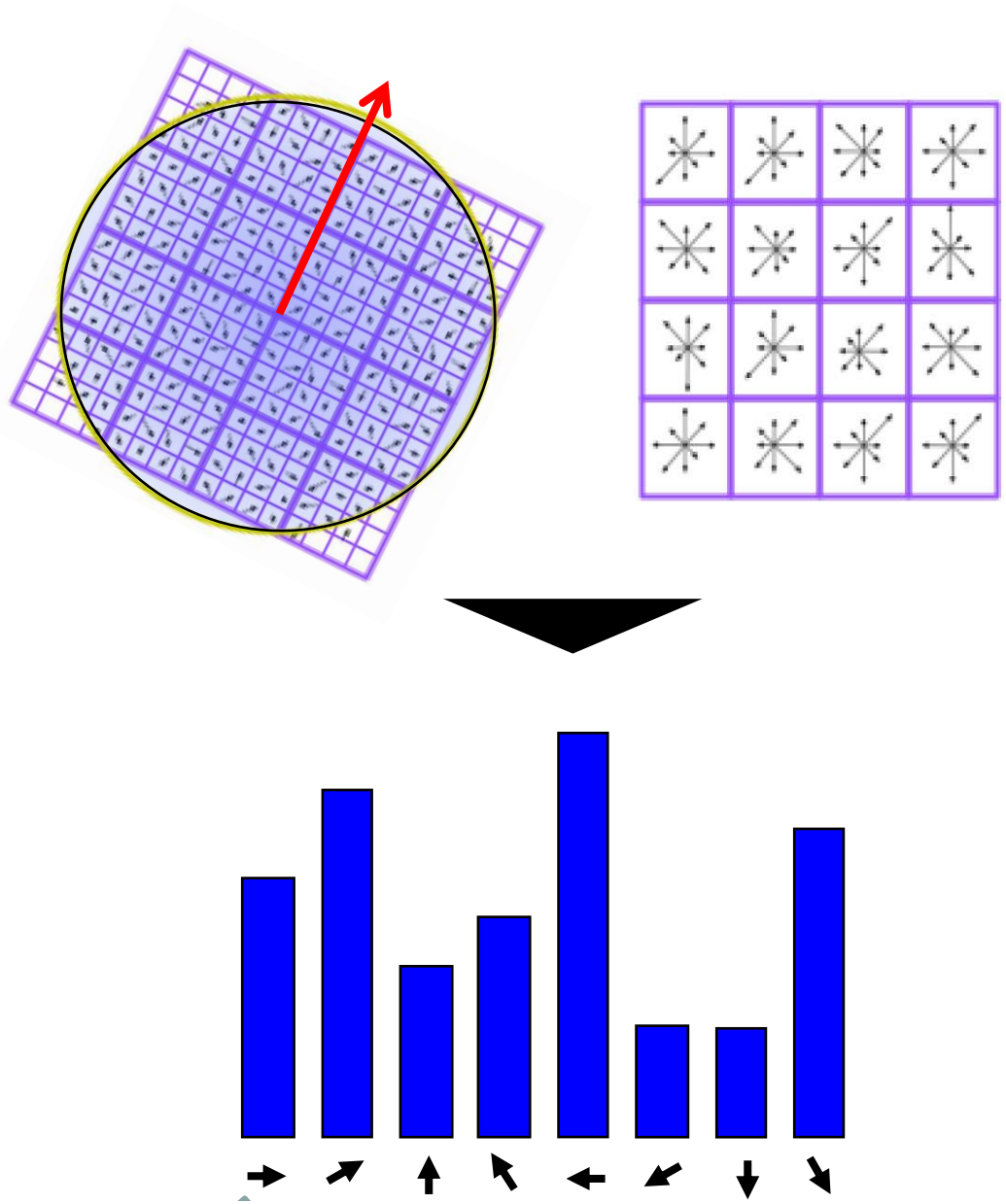


SIFT - Feature description

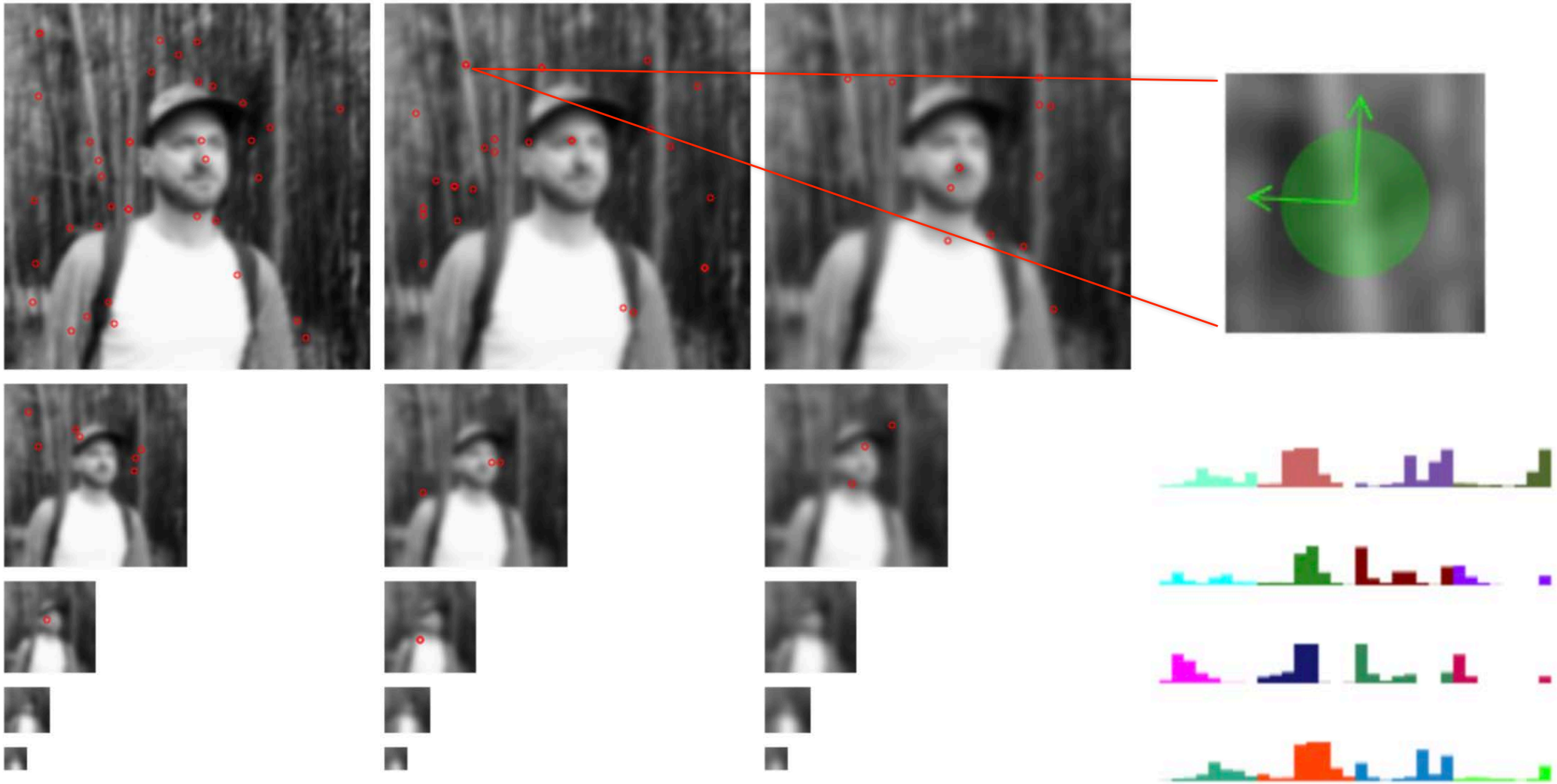


SIFT - Feature description from gradients

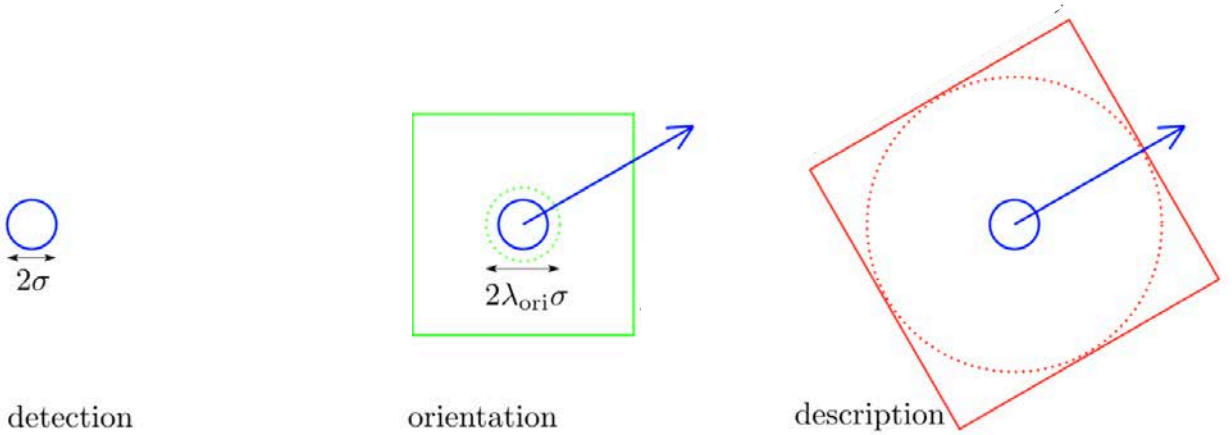
- Feature description (orientation and gradient)
  - Reorient image patch using main orientation
  - Create 16 gradient histograms (8 bins) weighted by magnitude and Gaussian window
  - Final keypoint Descriptor - 128 (4x4x8) element vector

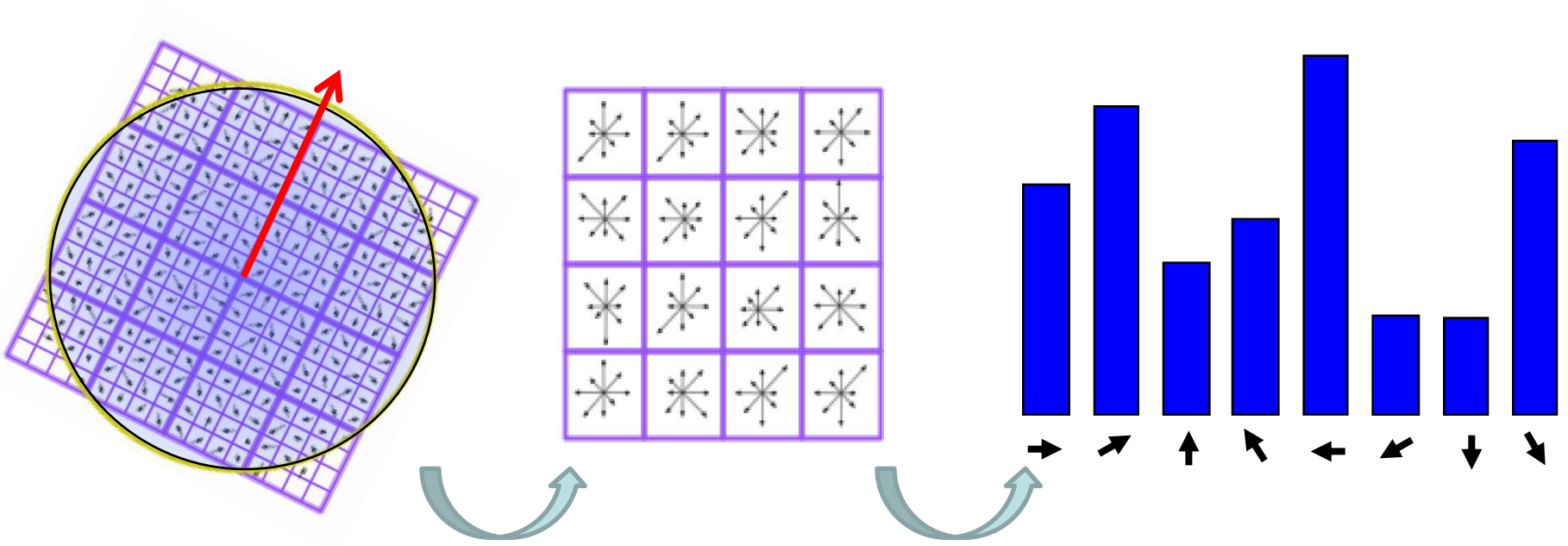


SIFT - Feature description from gradients



SIFT

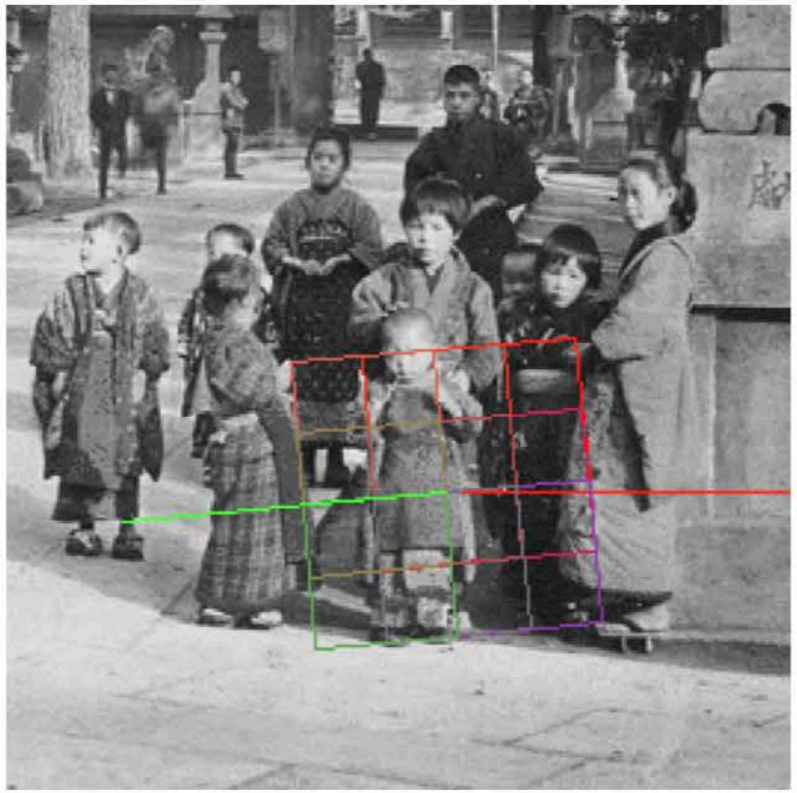

2σ
$2\lambda_{\text{ori}}\sigma$
detection
orientation
description

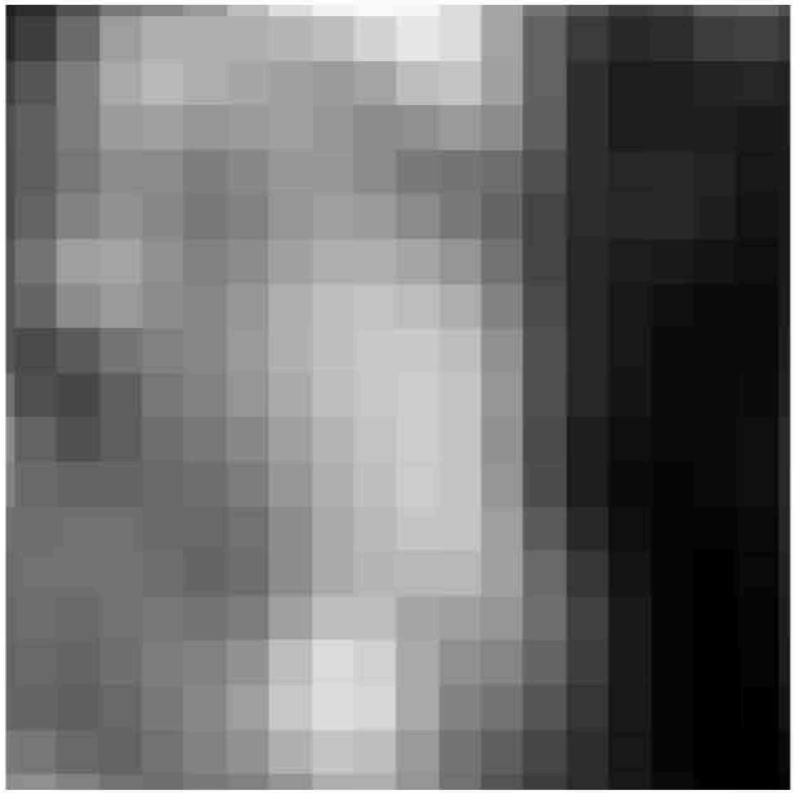


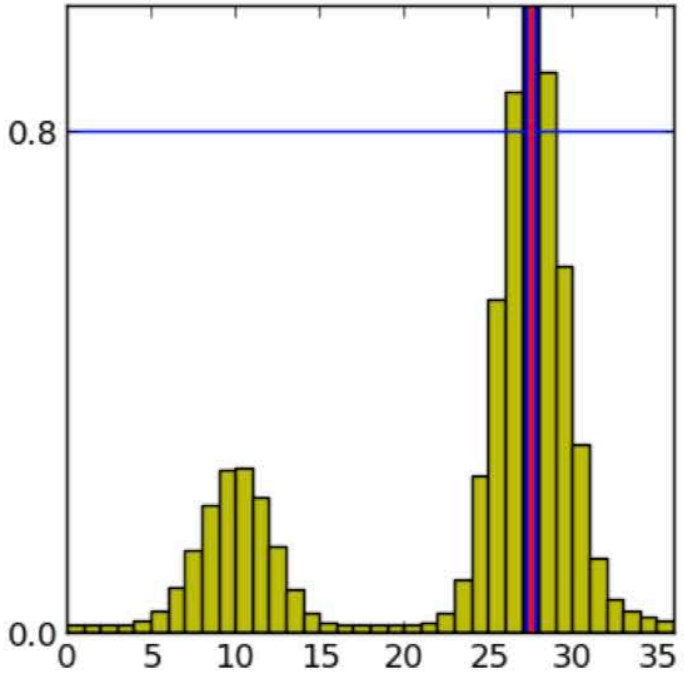
SIFT

Otero, I. R. (2015). *Anatomy of the SIFT Method* (Doctoral dissertation, École normale supérieure de Cachan-ENS Cachan).

SIFT

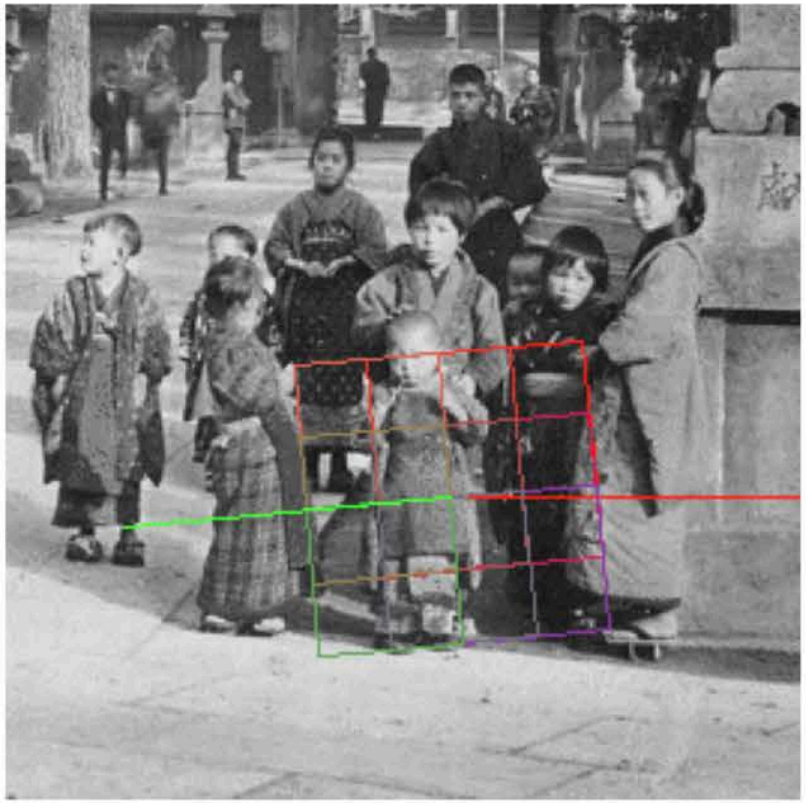


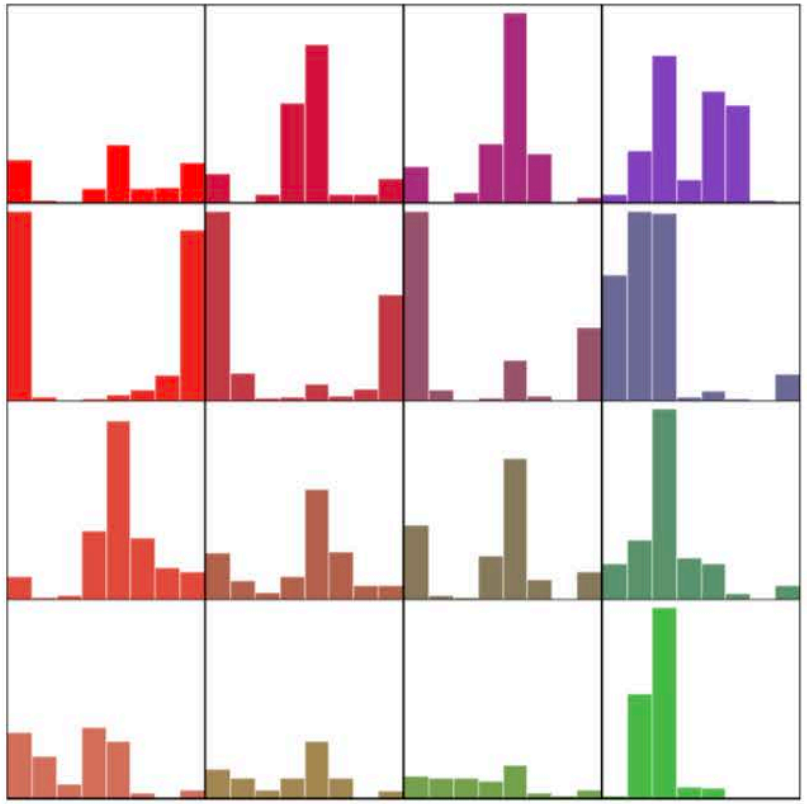




Otero, I. R. (2015). *Anatomy of the SIFT Method* (Doctoral dissertation, École normale supérieure de Cachan-ENS Cachan).

SIFT

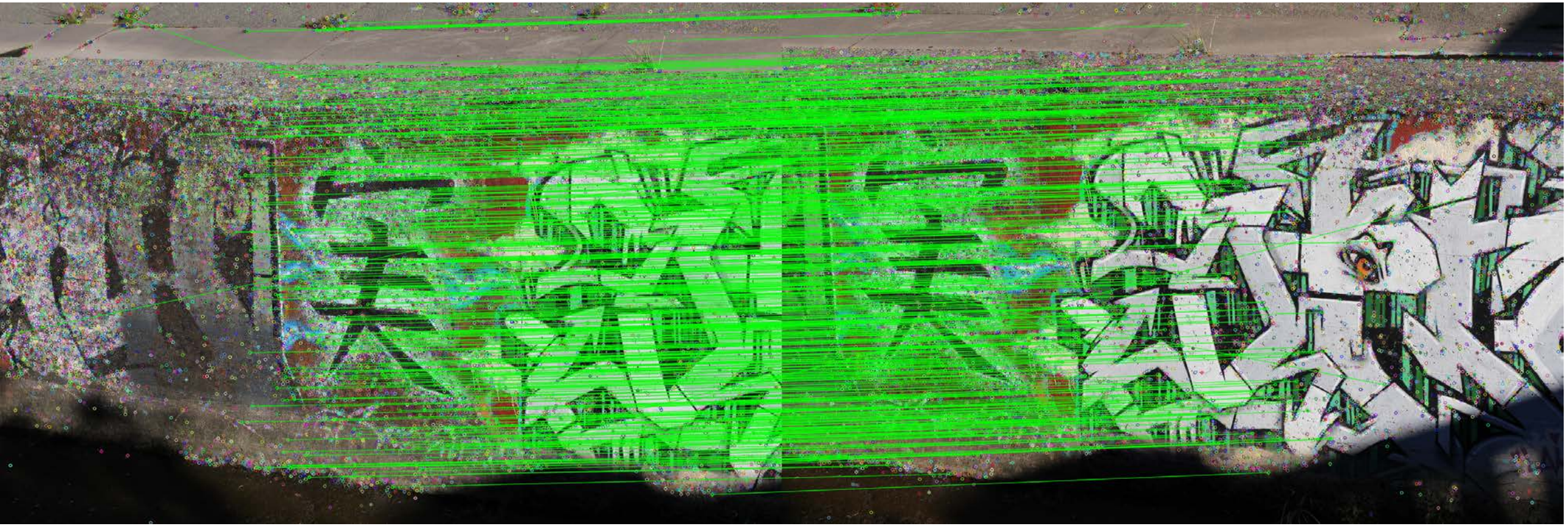




Otero, I. R. (2015). *Anatomy of the SIFT Method* (Doctoral dissertation, École normale supérieure de Cachan-ENS Cachan).

SIFT - Feature matching

SIFT - Matching



SIFT - Matching

- SIFT descriptor 128 values
  - Usually these are bytes
- Computing the distance between features:

$$d = \sqrt{\left(a_1 - b_1\right)^2 + \left(a_2 - b_2\right)^2 + \ldots}$$

  - 128 subtractions
  - 128 multiplications
  - 127 additions
  - 1 square root (optional)

SIFT - Matching

- SIFT descriptor 128 values
  - Usually these are bytes
- Computing the distance between features:

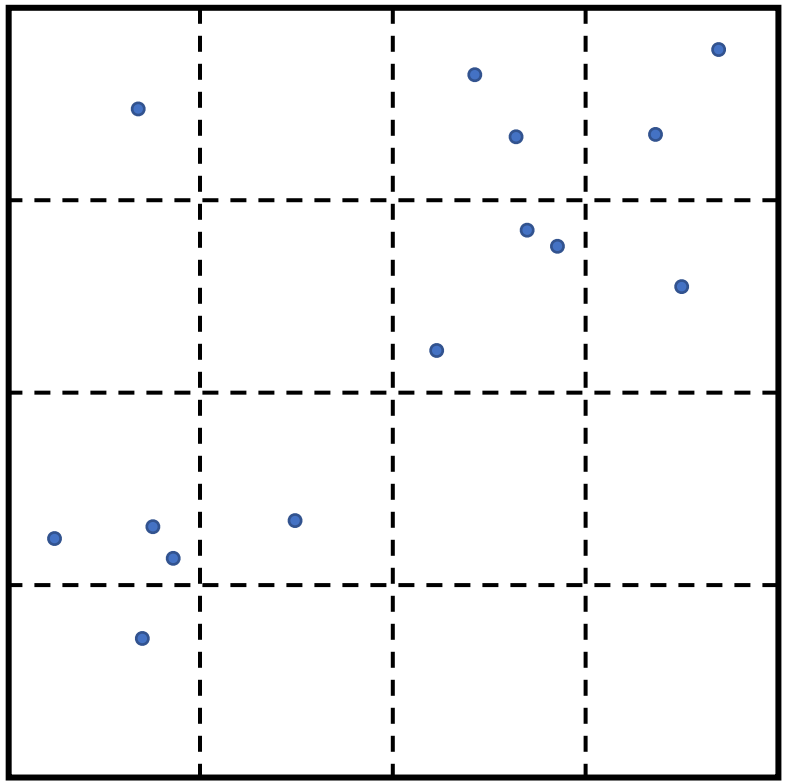
$$d = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \dots}$$

  - 128 subtractions
  - 128 multiplications
  - 127 additions
  - 1 square root (optional)

- Suppose we have 10,000 features in each image
  - Brute force method – find distance for each pair
  - Match is smallest distance
  - Matching one feature: $3{,}840{,}000$ operations
  - Matching all features: $38{,}400{,}000{,}000$ ops

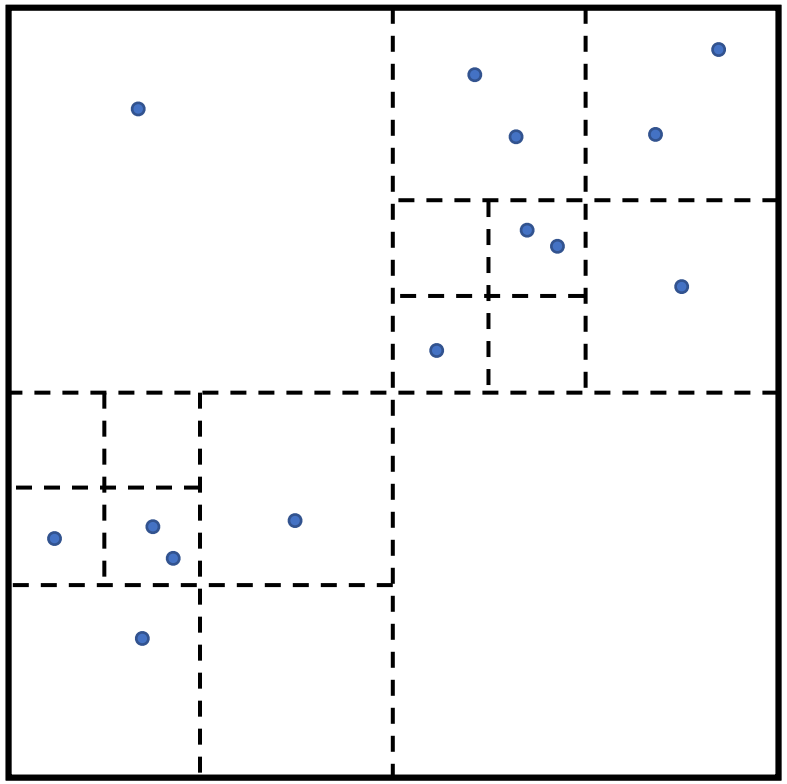
SIFT - Approximate Nearest Neighbours

- Split space into smaller regions
  - Each feature lies in one region
  - Only look for matches in same region
  - Might not find nearest match (why?)
- Uniform subdivision:
  - Divide space into a regular grid
  - 10 divisions on 2-D grid – 100 regions
  - 10 divisions on 128-D grid – 10128



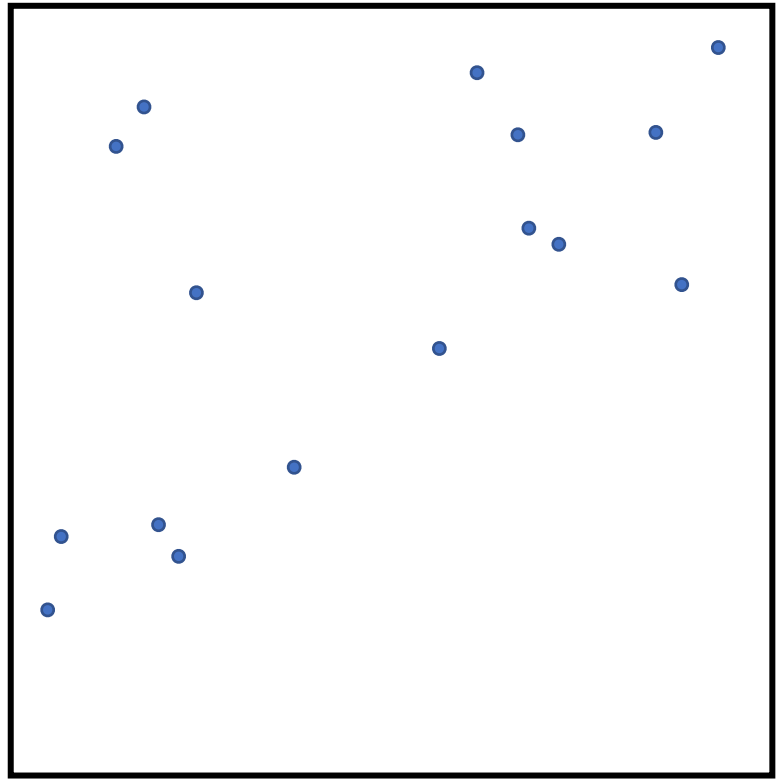
SIFT - Quadtrees, Octrees, etc.

- Recursively split space in half along each dimension
- Stop splitting when:
  - Some max depth is reached
  - Only a few items in the cell
- 2-D gives a quadtree
- 3-D gives an octree
- 128-D: : still too big
- 10 divisions on 128-D grid – 10128

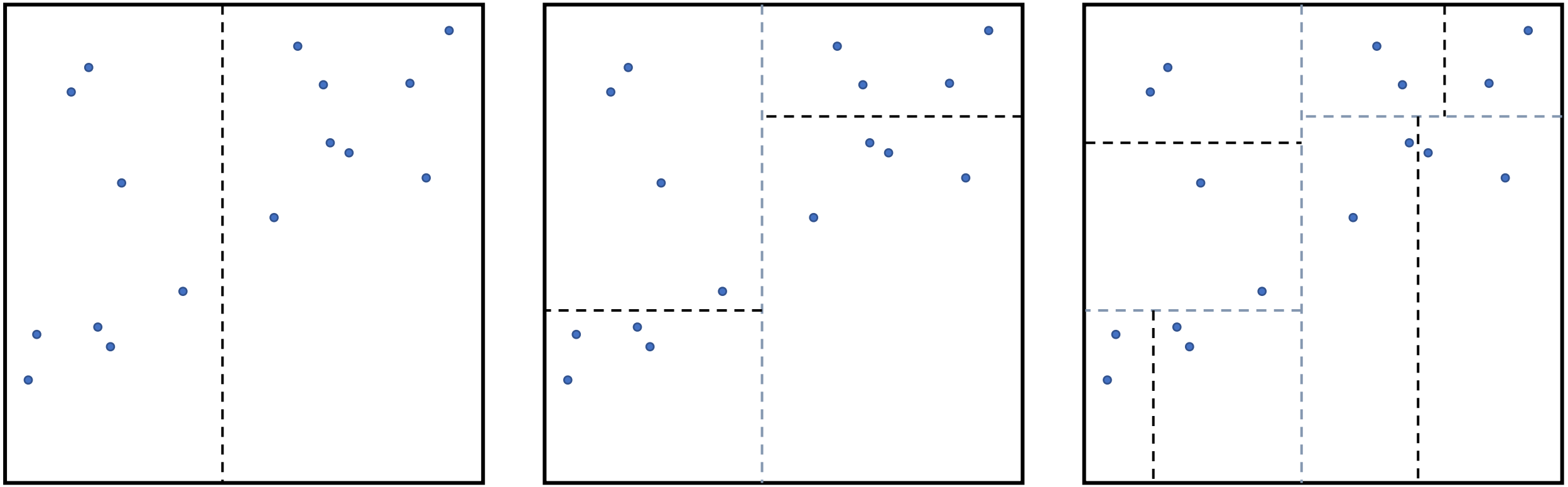


SIFT - K-D Trees

- Choose an axis and split along it
- Then take each half and repeat
- Stop when:
  - Only a few items in the region
  - Reached some maximum depth
- How to choose splits?
  - Axis – next, random, greatest spread?
  - Splitting point – middle or even split?



SIFT - K-D Trees



SIFT - K-D Trees and Feature Matching

- Put the feature descriptors from one image in a k-d tree
  - This has some overhead, but worth it if many features
- Given a feature from the other image:
  - Find what cell in the k-d tree contains that feature's descriptor
  - Compute the distance to all features in that cell
  - The nearest one is (probably) the best match
- For a tree with n layers and F features, this requires:
  - n comparisons to find the appropriate cell
  - $F/2^n$ descriptor-distance computations to find the best match

SIFT - Matching SIFT Features

- Even with brute-force, most SIFT matches are wrong
  - Lack of texture, repeating features, etc. -> ambiguous matches
  - And SIFT is the best we have
- With k-d trees this gets worse, but not much
- Solution – find best two matches
  - Ambiguous matches will have two similar distances
  - Keep only matches with a big difference between the two
  - Find two nearest matches $d\left(f, m_1\right) < \alpha d(f, m_2)$, $d(f, m_i)$ is distance to ith match, $\alpha < 1$ is a parameter, e.g.: 0.8
- Makes things better, but still some wrong matches

SIFT - Matching SIFT Features

- Even with brute-force, most SIFT matches are wrong
  - Lack of texture, repeating features, etc. -> ambiguous matches
  - And SIFT is the best we have
- With k-d trees this gets worse, but not much
- Solution – find best two matches
  - Ambiguous matches will have two similar distances
  - Keep only matches with a big difference between the two
  - Find two nearest matches $d\left(f, m_1\right) < \alpha d(f, m_2)$, $d(f, m_i)$ is distance to ith match, $\alpha < 1$ is a parameter, e.g.: 0.8
- Makes things better, but still some wrong matches

SIFT - Summary

SIFT - Matching SIFT Features

- Why does this work?
  - Key points are extracted at different scales and blur levels and all subsequent computations are performed within the scale space framework
    - **Invariance to image scaling and small changes in perspective**
  - Descriptor computation is relative to a reference (main) orientation
    - **Descriptors robust against rotation**
  - Descriptor information is stored relative to the key point position
    - **Invariance against translations**
  - Many potential key points are discarded if they are deemed unstable or hard to locate precisely
    - **Remaining key points should thus be relatively immune to image noise**
  - Histograms are normalized at the end which means the descriptors will not store the magnitudes of the gradients, only their relations to each other
    - **Descriptors are invariant against global, uniform illumination changes**
  - The histogram values are also thresholded to reduce the influence of large gradients. This will make the information partly immune to local, non-uniform changes in illumination.

The end!