Nonlinear Filters, Filter Combinations, and Padding

Machine Vision

Dr. Mostapha Kalami Heris

m.k.heris@shu.ac.uk

School of Engineering and Built Environment

Sheffield Hallam University

Learning Outcomes

By the end of this lecture, you should be able to:

- **Explain** why and when **nonlinear filters** are preferred over linear ones.
- **Apply** and **interpret median and related nonlinear filters** for noise removal and edge preservation.
- **Analyze** and **compare** common **padding and boundary handling** techniques.
- **Evaluate** which **filter types or combinations** best fit different image-processing goals.
- **Design** simple **filter pipelines** by combining linear and nonlinear filters in series or parallel.

Review of Last Lecture: Linear Filters and Kernels

- Filtering uses a **neighborhood operation** on each pixel.
- **Linear filters** combine pixels using **weighted sums**.
- Two main operations:
  - **Correlation** ($\star$) → kernel applied as-is
  - **Convolution** ($*$) → kernel flipped before application
- Common linear kernels:
  - **Identity**, **Box Blur**, **Sharpening**, **Sobel X/Y**

🧠 **Key takeaway:** Linear filters are powerful but can blur edges and amplify noise.

From Linear to Nonlinear Filters

Think about these.

> What happens if one pixel in the neighborhood is an extreme outlier (e.g., salt-and-pepper noise)?

> Will a linear average or a median be more robust?

> How might filtering behave at the image borders?

👉 These questions lead us to **nonlinear filters** and **padding strategies.**

Choosing the Right Filter

Group Activity: Filter–Application Matching

Work in **small groups (3–4 students)** to decide which filter best fits each real task.

Your sheet lists applications such as:

- Noise removal
- Edge detection
- Texture enhancement
- Medical image smoothing
- Artistic effects

🕒 **Time:** 5–7 minutes

🗣️ Discuss, choose a suitable **filter type** (linear / nonlinear) and **specific filter name**, then justify your reasoning.

Debrief & Discussion

🔍 Which filters did your group assign to each application?

💬 What patterns or conflicts appeared?

💡 How did you decide between linear and nonlinear approaches?

Let’s compare answers.

Example Applications and Suitable Filters

Application	Goal	Typical Filter(s)	Linear / Nonlinear
Noise reduction (Gaussian noise)	Smooth variations	Gaussian blur	Linear
Salt & pepper noise	Remove outliers	Median	Nonlinear
Edge detection	Find object boundaries	Sobel, Prewitt	Linear
Texture enhancement	Sharpen fine details	Laplacian + Sharpen	Linear
Medical imaging	Preserve edges, remove impulse noise	Median + Bilateral	Nonlinear

🧭 **Key insight:** Each filter family suits specific tasks — no single "best" filter fits all situations.

Nonlinear Filters

Let's Think About Nonlinear Filters

These are some questions for you. Think about them for a moment.

> **Q1:** Can you think of a filtering operation that is not linear?
>
> **Q2:** Why might we want to use nonlinear filters instead of linear ones?
>
> **Q3:** What are some examples of nonlinear filtering operations?

🗣️ **Let's discuss.**

What do you think?

Can you come up with any answers?

Discussion Summary

Some examples of nonlinear filters include:

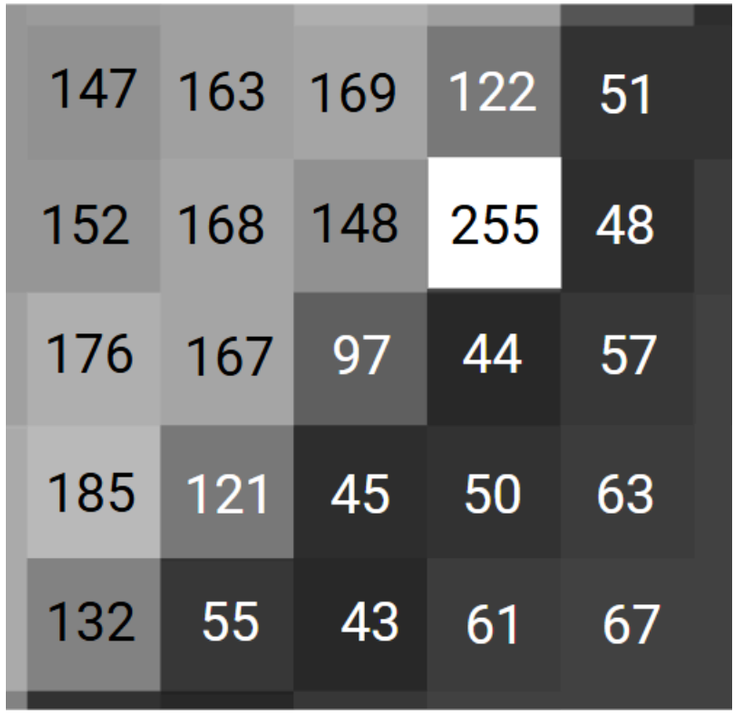
- **Median filter:** Replaces each pixel with the median value of its neighborhood.
- **Mode filter:** Replaces each pixel with the most frequently occurring value in its neighborhood.
- **Maximum/Minimum filters:** Replace each pixel with the maximum or minimum value in its neighborhood.

Why use nonlinear filters?

- They can be more effective at removing certain types of noise (e.g., salt-and-pepper noise).
- They can preserve edges better than linear filters.
- They can handle outliers more robustly.

An Example: Median Filter

The median filter replaces each pixel with the median value of its neighborhood. Let's apply a 3x3 median filter to the following image.



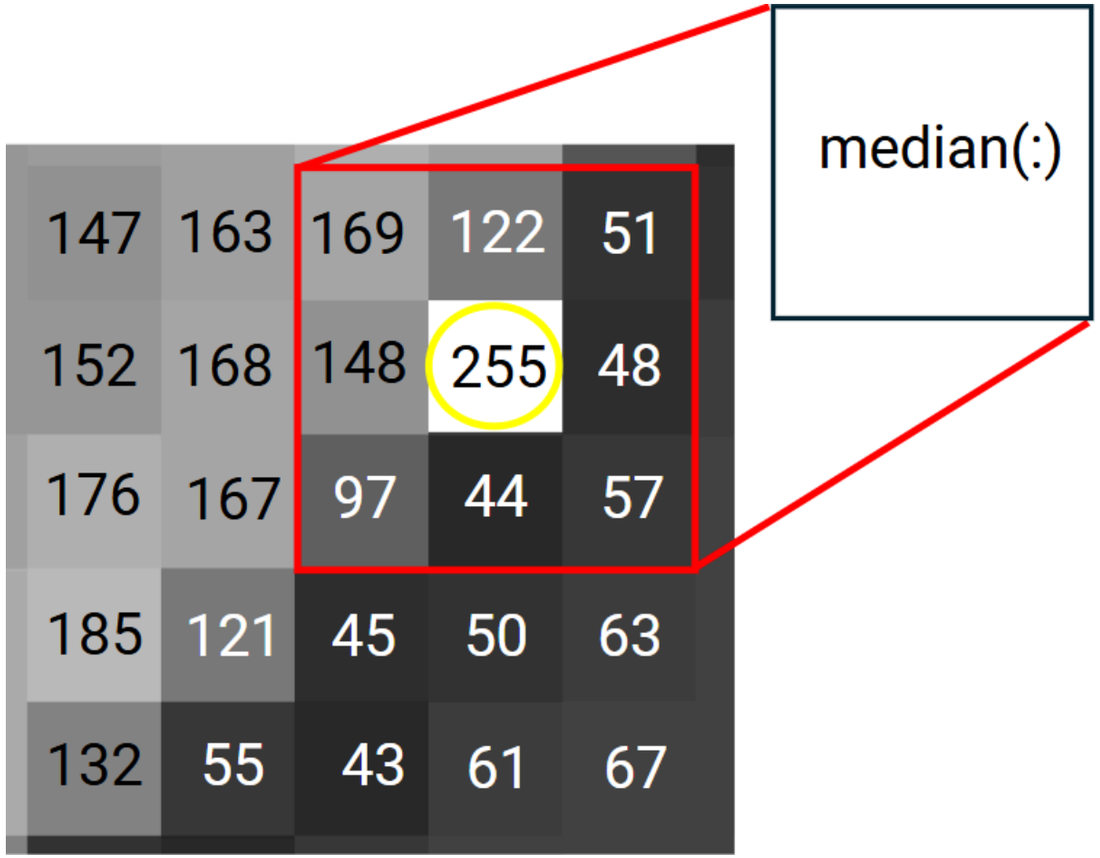
Median Filter Operation

Like linear filters, we consider a 3x3 window around each pixel and move it across the image.

For each position of the window, we find the median value of the pixels within the window.

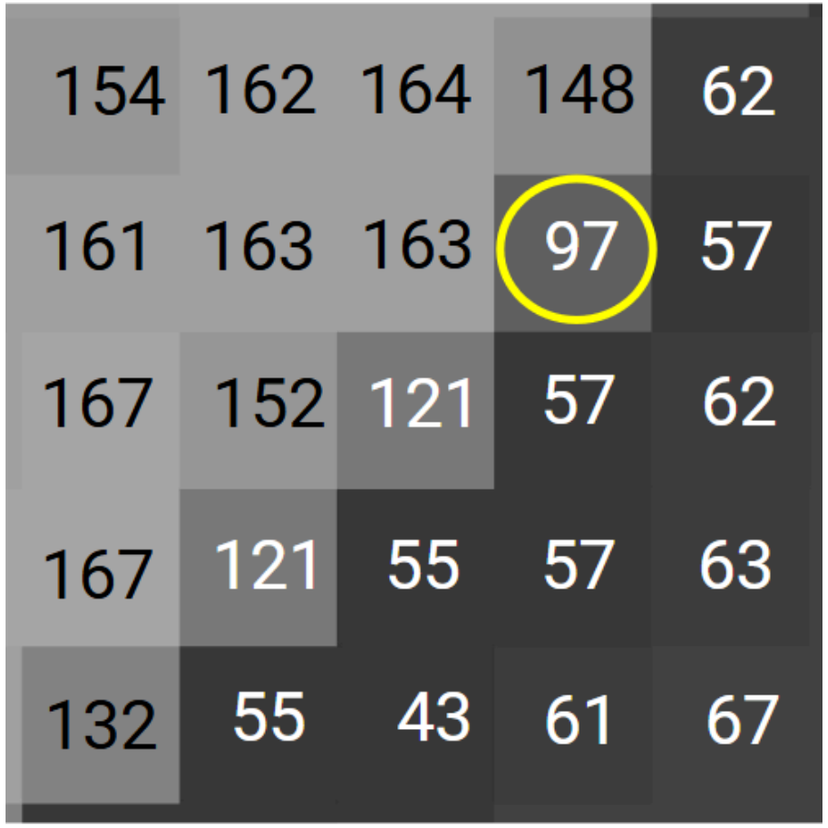
This the median value of the pixels in the 3x3 window:

$$\text{Output} = \text{median}\left\{\begin{matrix} 169 & 122 & 51 \\ 148 & 255 & 48 \\ 97 & 44 & 57 \end{matrix}\right\} = 97$$

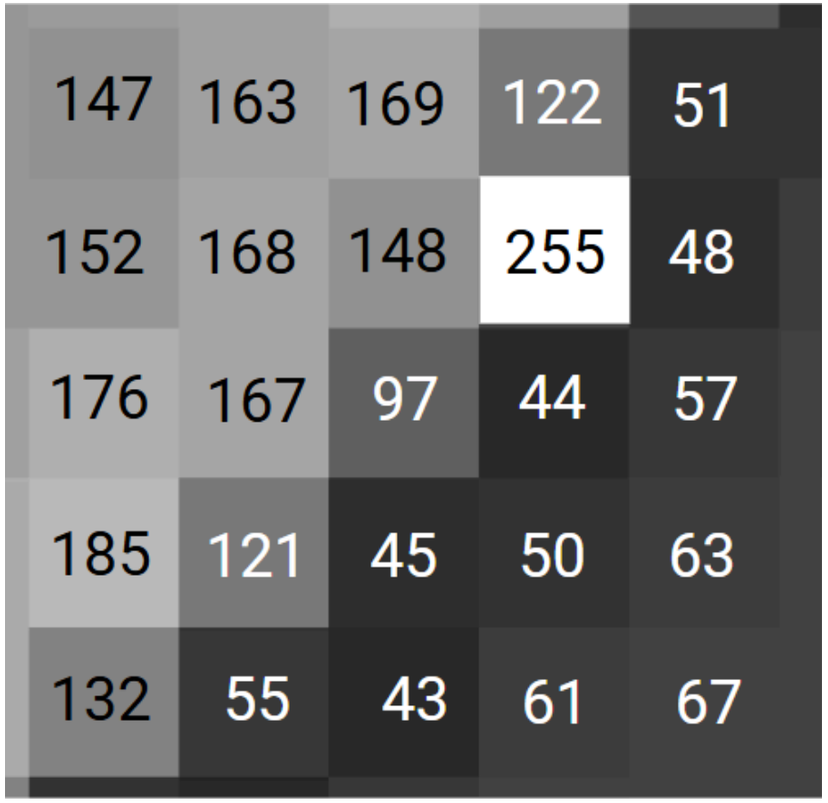


Final Output of Median Filter

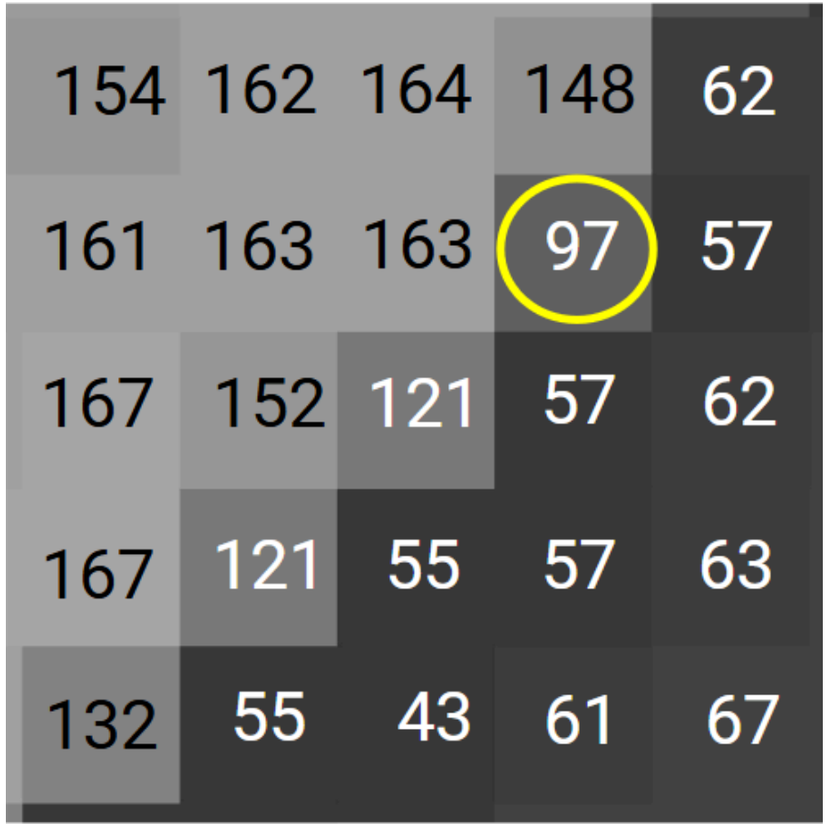
After applying the median filter to the entire image, we get the following output.



Median Filter: Input vs. Output



Input Image



Output Image

Sheffield Hallam University

MATLAB Example: Median Filtering

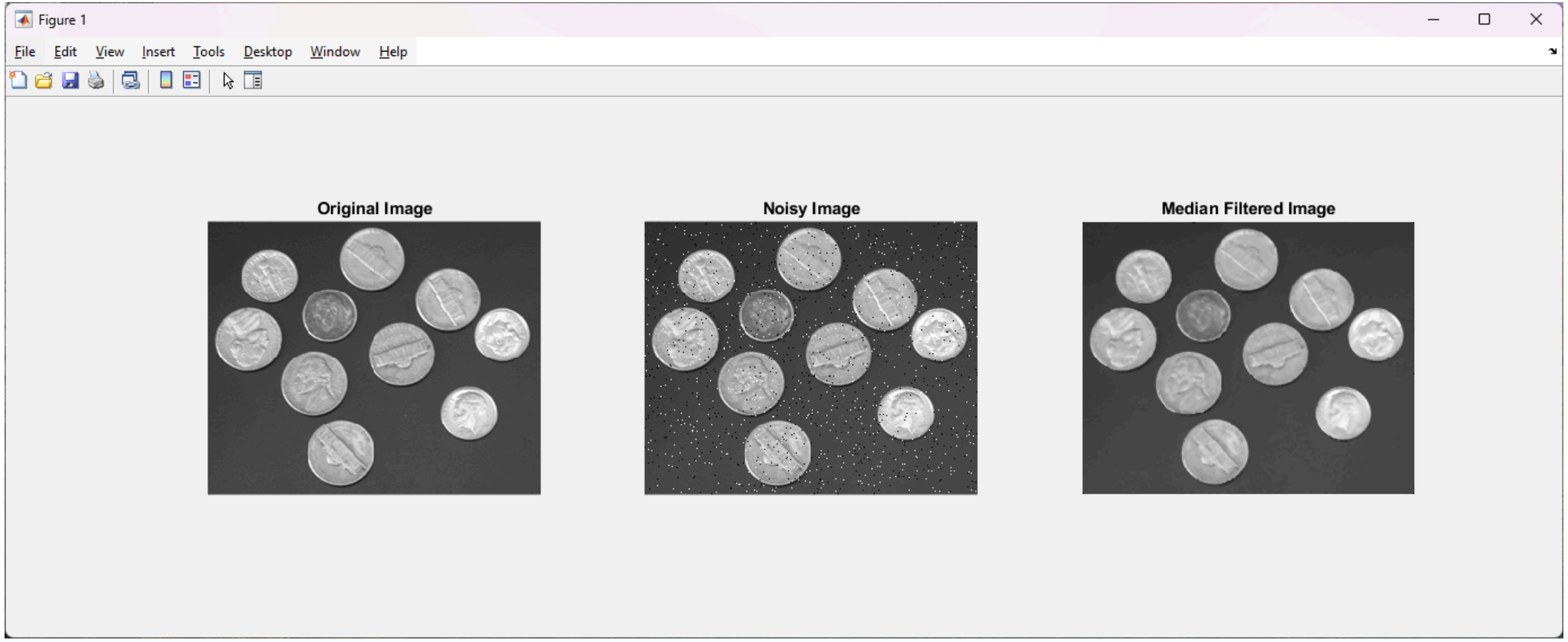
```MATLAB
% Read the image
I = imread('coins.png');

% Add salt & pepper noise to the image
J_noisy = imnoise(I, 'salt & pepper', 0.02);

% Apply median filtering
K = medfilt2(J_noisy, [3 3]);

% Display the original, noisy, and filtered images
subplot(1,3,1), imshow(I), title('Original Image');
subplot(1,3,2), imshow(J_noisy), title('Noisy Image');
subplot(1,3,3), imshow(K), title('Median Filtered Image');
```

Result of the MATLAB Code



Sheffield Hallam University

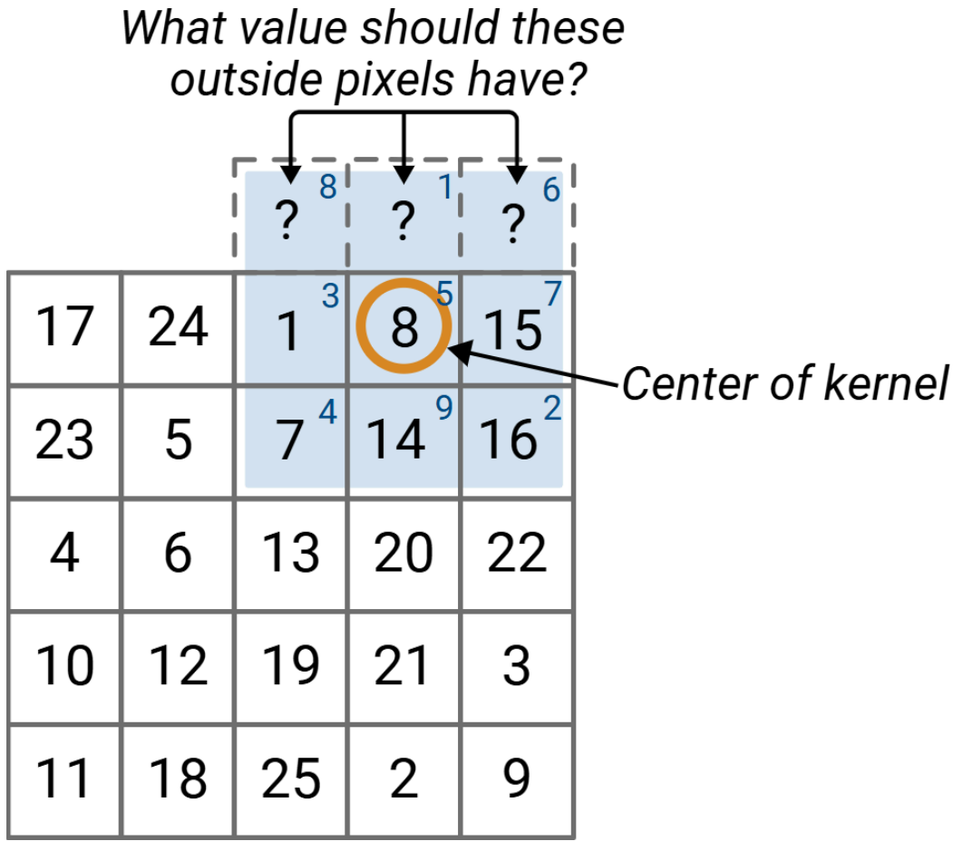
Padding and Boundary Handling

Why Padding is Needed?

For pixels near the border, the filter window may extend beyond the image boundaries.

To address this, we can use **padding** techniques to extend the image.

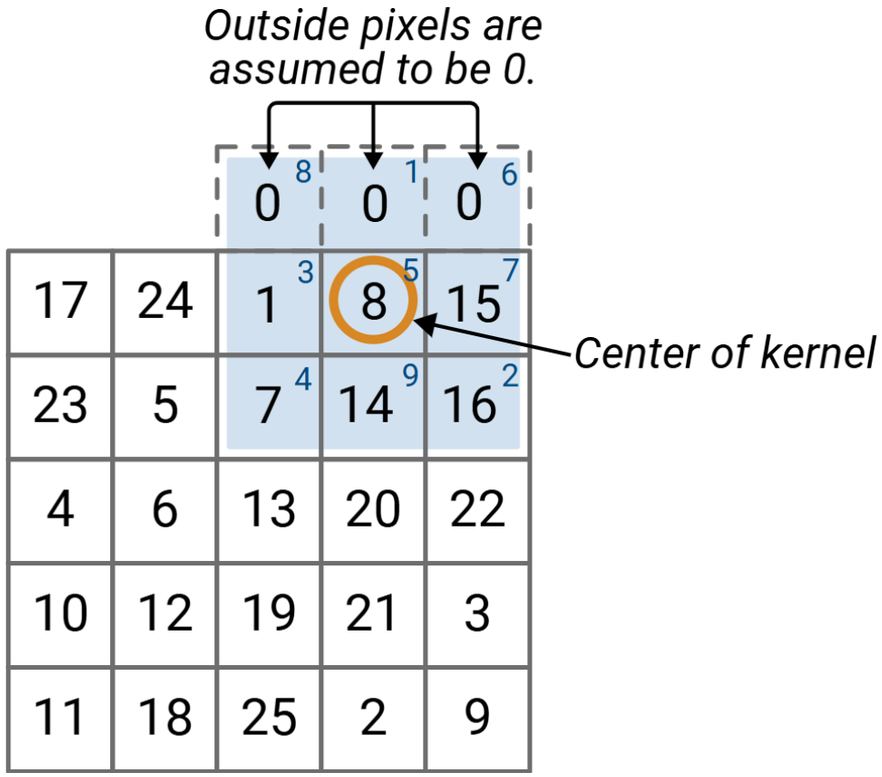
> **How can we handle image boundaries?**
>
> Do you have any ideas?



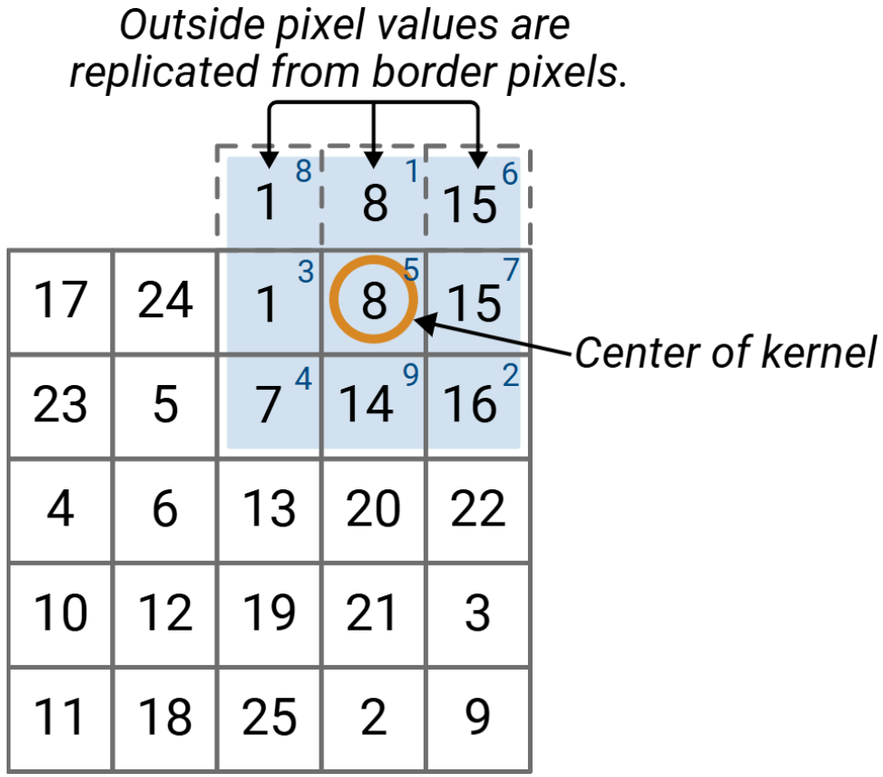
Sheffield Hallam University

Two Common Padding Methods

Zero Padding: Add zeros around the image borders.

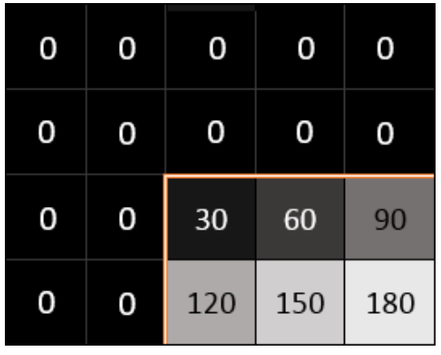


*__Replication Padding:__ Extend the edge pixels outward.



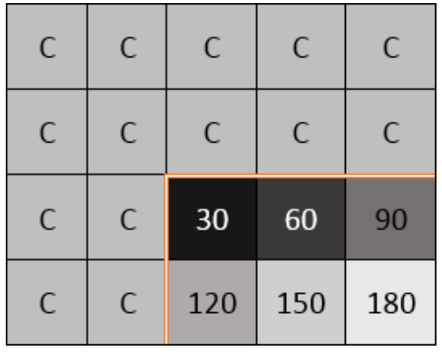
Sheffield Hallam University

Common Padding Options



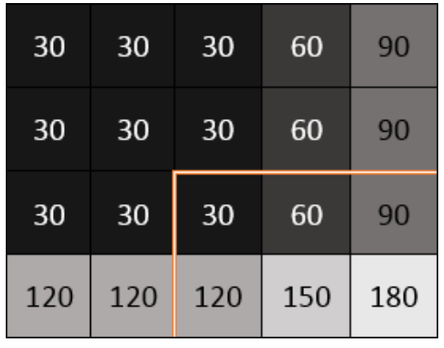
Zero Padding

Fill with zeros



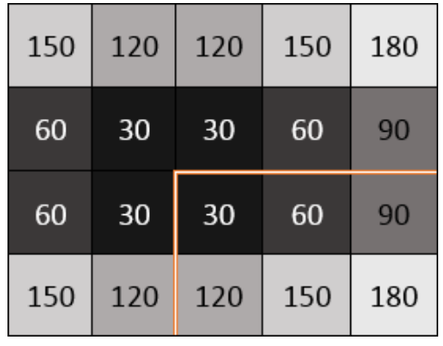
Constant Padding

Fill with a constant value



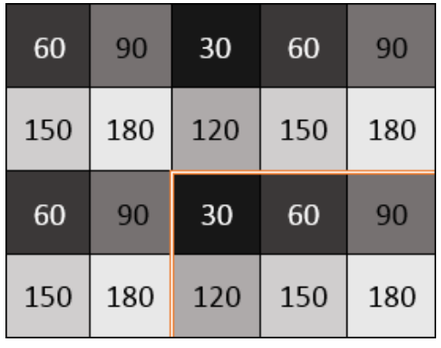
Replication Padding

Replicate edge pixels



Symmetric Padding

Mirror the image at the borders



Circular Padding

Wrap around the image

Combining Filters

Why Combine Filters?

- Some image-processing goals need **more than one filter**.
- Filters can be connected in:
  - **Series:** apply one, then another (output → next input)
  - **Parallel:** apply both, then combine the results
- Combining filters lets us design custom effects and improve robustness.

> 💡 Think of filters as “building blocks” for image-processing pipelines.

Example: Sobel X + Sobel Y → Omnidirectional Edges

- **Sobel X** detects **vertical** edges.
- **Sobel Y** detects **horizontal** edges.
- Combine their magnitudes to get edges in **all directions**.
- You can also compute the **edge direction**.

Formula:

$$G = \sqrt{(G_x^2 + G_y^2)}$$

$$\theta = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

MATLAB Code

```matlab
hx = fspecial('sobel');    % Sobel X kernel
hy = hx';                  % Sobel Y kernel
Gx = imfilter(I, hx);      % Sobel X response
Gy = imfilter(I, hy);      % Sobel Y response
G = sqrt(Gx.^2 + Gy.^2);   % Combine magnitudes
P = atan2(Gy, Gx);         % Edge directions
imshow(G, []);
```

> The combined result is more complete than either filter alone.

More Combination Ideas

- **Series combinations**
  - Median → Sobel: reduce noise, then detect edges
  - Gaussian → Laplacian: smooth, then sharpen (LoG)
- **Parallel combinations**
  - Apply two filters and **merge outputs**, e.g. averaging or weighting

> What happens if you apply sharpening first, then median filtering? Which sequence produces cleaner results?

> 🗣️ **Discuss:** How could combining filters help in your own research or applications?

Summary & References

Summary of Key Points

- **Linear filters** use weighted sums of neighboring pixels.
- **Nonlinear filters** (e.g., median) rely on order statistics for better edge and noise handling.
- **Padding** ensures every pixel can be processed near borders.
- **Choosing the right filter** depends on the **task goal** — denoising, sharpening, or edge detection.
- **Combining filters** (in series or parallel) can produce stronger or more tailored results.
- MATLAB offers flexible functions: `imfilter`, `medfilt2`, and custom pipelines.

> 🧭 Think of filtering as a *design space* — not just a toolbox.

Reflect and Design: Thinking About Filter Strategies

💭 **Reflect before next time:**

> **Q1:** How would you decide between a linear and a nonlinear filter for a new application?

> **Q2:** What sequence of filters might handle both noise and edge preservation?

> **Q3:** Can you design a simple pipeline combining at least two filters for a task you care about?

Next week: We'll shift from **spatial-domain** to **frequency-domain filtering**. Seeing images in an entirely new way.

References

- Gonzalez, R. C., & Woods, R. E. (2018). *Digital Image Processing* (4th ed.). Pearson.
- Gonzalez, R. C., Woods, R. E., & Eddins, S. L. (2020). *Digital Image Processing Using MATLAB* (3rd ed.). Gatesmark Publishing.
- Forsyth, D. A., & Ponce, J. (2011). *Computer Vision: A Modern Approach* (2nd ed.). Pearson.
- Szeliski, R. (2010). *Computer Vision: Algorithms and Applications*. Springer.
- MathWorks. (n.d.). *Image Processing Toolbox – MATLAB*. Retrieved from https://mathworks.com/help/images/index.html

Exit Ticket

Before you leave, please complete the online form and provide your feedback.

1. **One clear concept** from today
2. **One confusing concept**
3. **One thing to try in MATLAB** this week

Scan the QR code or visit the link below.

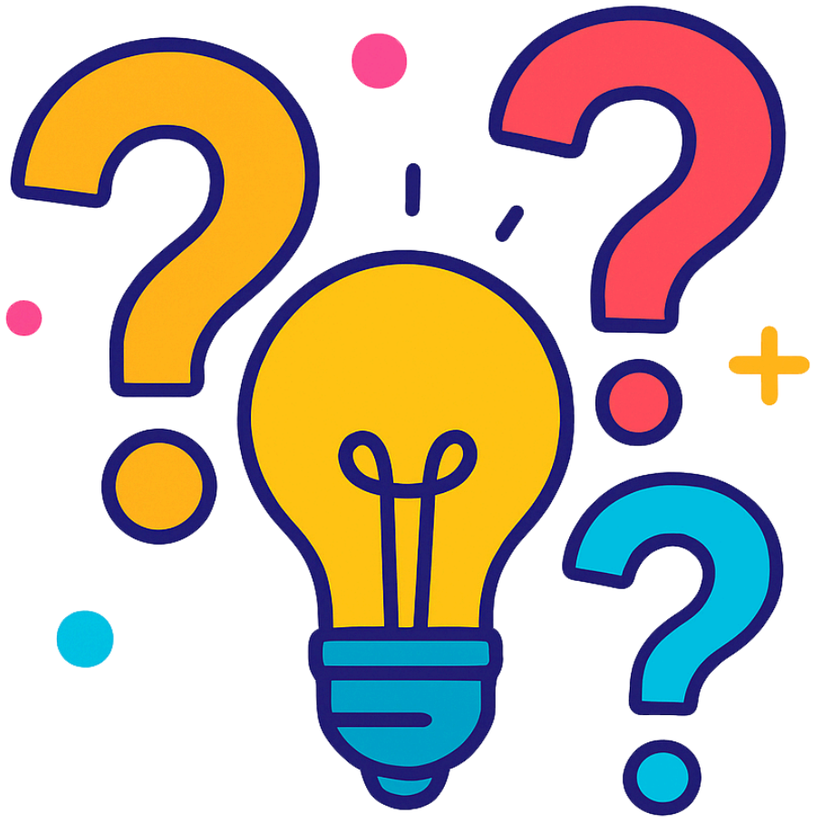
forms.office.com/e/YvWnAr66yJ

Sheffield Hallam University

Questions & Support

- You are encouraged to **ask questions anytime** 💡
  - During the lecture
  - In lab sessions
  - By email → m.k.heris@shu.ac.uk
- **No question is too simple** — asking questions:
  - Helps you learn faster
  - Builds confidence
  - Improves understanding for the whole class

👉 If something is unclear, **just ask!**



Sheffield Hallam University