Lost in Pieces:
An Image Jigsaw Reconstruction Challenge

Dionisios N. Sotiropoulos

December 16, 2025

Outline I

Outline II

Introduction

Context. In many computer vision problems, images are not given as neat, fully observed entities.

Instead, they often appear as:

- local patches,
- segmented regions,
- partially occluded objects.

Challenge. Inferring the underlying image or scene from such incomplete evidence is demanding:

- requires robust feature extraction,
- requires careful reasoning about spatial and contextual relationships.

Jigsaw Puzzle Perspective

This project adopts a *jigsaw puzzle* view of the problem.

Setup.

- A given image is decomposed into pieces (*tiles* or *regions*).
- The pieces are shuffled.
- The pieces may also be rotated.

Task. Reconstruct the original arrangement of pieces using only visual cues.

From Classical to Deep Descriptors

You are expected not only to *apply* but also to *extend* feature extraction concepts from the course.

Descriptor families to combine:

- Classical descriptors:
  - region descriptors (e.g. area, moments),
  - boundary descriptors (shape, contour-based),
  - texture features (filter banks, histograms),
  - local interest point descriptors.
- Deep representations:
  - features derived from pre-trained CNNs,
  - used on tiles and/or their borders.

Project Goals

The project has a dual purpose:

- **Conceptual:** to consolidate and deepen your understanding of feature extraction and image descriptors.
- **Practical / design-oriented:** to introduce a realistic, open-ended problem where:
  - multiple competing design choices must be considered,
  - evaluation criteria must be explicitly defined, explored and justified.

Feature Extraction in Image Analysis

Core idea. Feature extraction maps raw image data into a compact, informative representation suitable for:

- recognition,
- classification,
- matching and retrieval.

Goal. Replace pixels with higher-level descriptors that:

- are robust to nuisance variations,
- remain discriminative for different regions and objects.

Types of Classical Descriptors

The course textbook presents several families of descriptors:

- **Region-based descriptors**
  - area, centroid, orientation,
  - compactness, geometric and invariant moments.
- **Boundary and shape descriptors**
  - contour-based signatures,
  - Fourier descriptors, curvature-based features.
- **Texture descriptors**
  - filter bank responses,
  - Gabor filters,
  - co-occurrence-based features.
- **Local interest point descriptors**
  - SIFT-like, SURF-like descriptors,
  - capture distinctive patterns in local neighborhoods.

Invariance and Robustness

Classical descriptors are designed to be:

- invariant (or at least robust) to:
  - translation,
  - rotation,
  - scale changes,
  - moderate illumination changes;
- while remaining *discriminative* for:
  - different regions of the same image,
  - different objects or classes.

Design tension:

- too much invariance $\Rightarrow$ descriptors become less distinctive,
- too little invariance $\Rightarrow$ descriptors become unstable.

Deep Learning-Based Image Representations

Deep convolutional neural networks (CNNs) learn powerful image representations.

Hierarchical features:

- early layers: edges, corners, simple textures,
- intermediate layers: motifs, parts, repeated patterns,
- deeper layers: object- and scene-level semantics.

These learned features often outperform hand-crafted descriptors in many tasks.

CNNs as Generic Feature Extractors

For this project, you are encouraged to use *pre-trained* CNNs as feature extractors.

Typical pipeline:

- input: image patch or border strip,
- forward pass through a pre-trained network,
- take intermediate feature maps (activations),
- apply spatial pooling to obtain a fixed-length vector.

Advantages:

- strong invariance to small translations, deformations, illumination changes,
- complementary to classical descriptors (color, texture, shape),
- can be used for both tiles and side-level border regions.

Project Objectives

The main objective of the project is to design, implement, and evaluate a system that reconstructs a segmented image from its shuffled pieces.

1. **Feature Extraction**
2. **Deep Features**
3. **Adjacency Modeling**
4. **Global Reconstruction**
5. **Evaluation**

Image Segmentation and Puzzle Pieces

Let the original image be a function

$$I : \Omega \to \mathbb{R}^C, \qquad \Omega = \{1, \dots, H\} \times \{1, \dots, W\},$$

where:

- H, W are the height and width in pixels,
- C is the number of channels (e.g. $C = 3$ for RGB).

Fix integers $P, Q \in \mathbb{N}$ such that $P \mid H$ and $Q \mid W$. Define

$$h = \frac{H}{P}, \qquad w = \frac{W}{Q},$$

so that each tile has size $h \times w$ pixels.

Grid and Tiles

The set of grid positions is

$$\mathcal{G} = \{(r, c) : 1 \leq r \leq P,\ 1 \leq c \leq Q\}.$$

For each $(r, c) \in \mathcal{G}$, the corresponding tile (region) is

$$R_{r,c} = \big\{(x, y) \in \Omega : h(r-1) + 1 \leq x \leq hr,\ w(c-1) + 1 \leq y \leq wc\big\}.$$

These tiles form a partition of the image domain:

$$\Omega = \bigcup_{(r,c)\in\mathcal{G}} R_{r,c}, \qquad R_{r,c} \cap R_{r',c'} = \emptyset \text{ for } (r, c) \neq (r', c').$$

Indexing the Tiles

Let

$$N = PQ$$

be the total number of tiles.

For convenience, index the tiles as

$$\mathcal{R} = \{R_1, \ldots, R_N\},$$

where the index k is associated with a unique grid position $(r_k, c_k) \in \mathcal{G}$ via a fixed bijection

$$g^\star : \{1, \ldots, N\} \to \mathcal{G}, \qquad g^\star(k) = (r_k, c_k).$$

Puzzle Generation

During puzzle generation, the tiles $\{R_k\}$ are transformed as follows:

- **Shuffling:** A permutation σ of $\{1, \ldots, N\}$ is applied, so that the logical piece index k no longer corresponds to its original grid position $g^\star(k)$.
- **Rotation (optional):** For each tile R_k, a rotation angle

$$\theta_k^\star \in \mathcal{A}, \qquad \mathcal{A} = \{0^\circ, 90^\circ, 180^\circ, 270^\circ\},$$

  is chosen and applied.

You are given:

- the resulting set of tile images (after shuffling and rotation),
- the grid dimensions (P, Q).

Hidden Ground Truth

The ground-truth information

$$g^\star : \{1, \ldots, N\} \to \mathcal{G}, \qquad \{\theta_k^\star\}_{k=1}^N,$$

is stored internally during puzzle generation.

It is used at the end to:

- evaluate each reconstructed solution,
- compare it against the original image I in terms of placement and orientation.

Reconstruction Task

The reconstruction task consists of two intertwined subproblems:

1. **Piece Placement:** Estimate an assignment

$$\pi : \{1, \ldots, N\} \to \mathcal{G}$$

that places each observed piece at a unique position in the $P \times Q$ grid.

2. **Orientation Estimation (optional):** Estimate an orientation

$$\hat{\theta}_k \in \mathcal{A}$$

for each piece k, if rotations are allowed.

Ideal Reconstruction and Evaluation

The ideal solution would recover the hidden ground truth:

$$\pi(k) = g^\star(k) \quad \text{and} \quad \hat{\theta}_k = \theta_k^\star \quad \text{for all } k.$$

In practice, we aim to approximate this as closely as possible.

Because $g^\star$ and $\theta_k^\star$ are known to the puzzle-generation code but hidden from you, they provide a precise reference for quantitative evaluation of your reconstruction.

Geometry of Tiles and Border Strips

- The original image is a discrete function

$$I : \Omega \to \mathbb{R}^C, \qquad \Omega = \{1, \ldots, H\} \times \{1, \ldots, W\}.$$

- It is partitioned into N non-overlapping rectangular regions (tiles)

$$\{R_k \subset \mathbb{Z}^2\}_{k=1}^{N},$$

  each corresponding to one cell of a $P \times Q$ grid.
- The tiles form a partition:

$$\Omega = \bigcup_{k=1}^{N} R_k, \qquad R_k \cap R_\ell = \emptyset \text{ for } k \neq \ell.$$

Geometry of Tiles and Border Strips

- Each tile R_k has four sides, indexed by

$$\mathcal{S} = \{\text{top}, \text{right}, \text{bottom}, \text{left}\} = \{N, E, S, W\}.$$

- Fix an integer parameter $w_{\text{b}} \geq 1$ that specifies the width of the *border strip* around each side.
- For a tile R_k and a side $s \in \mathcal{S}$, the border strip $B_s(R_k)$ is the set of pixels of R_k that lie within w_{b} pixels from side s.

If the bounding box of R_k is

$$R_k = \{(x, y) : x_{\min} \leq x \leq x_{\max},\ y_{\min} \leq y \leq y_{\max}\},$$

then:

- $B_{\text{top}}(R_k)$: the w_{b} top-most rows of R_k,
- $B_{\text{bottom}}(R_k)$: the w_{b} bottom-most rows of R_k,
- $B_{\text{left}}(R_k)$: the w_{b} left-most columns of R_k,
- $B_{\text{right}}(R_k)$: the w_{b} right-most columns of R_k.

Geometry of Tiles and Border Strips

- Border strips $B_s(R_k)$ provide the natural geometric support for *side-level descriptors*.
- They capture the local appearance near a side where two tiles may potentially meet in the original image.
- Any region-based descriptor introduced later (color, texture, local, deep) can be evaluated:
  - either on an entire tile R_k (piece-level descriptor),
  - or on a side border strip $B_s(R_k)$ (side-level descriptor).
- This unified view allows us to reuse the same feature families at different geometric scales (piece vs. side) without redefining them.

Color and Texture Descriptors

- Goal: capture the *local appearance* of a region (piece or border strip) in a mathematically well-defined way.
- We will use:
  - **Color descriptors** (color histograms in a chosen color space),
  - **Texture descriptors** (based on filter-bank responses, discussed next).
- The same constructions apply:
  - on a whole tile R (piece-level descriptor),
  - or on a border strip $B_s(R)$ (side-level descriptor).

Color and Texture Descriptors

Color histograms.

- Let $I : \Omega \to \mathbb{R}^C$ be the image in some color space (e.g. RGB, HSV, Lab).
- Let $R \subset \Omega$ be a region (whole piece or border strip).

- Fix a number of bins $B \in \mathbb{N}$ per channel.
- For a single channel $c \in \{1, \dots, C\}$, let $I_c^{\min}, I_c^{\max}$ be the minimum and maximum values (e.g. 0 and 255) and define bin intervals

$$\mathcal{I}_b = \Big[I_c^{\min} + \frac{b-1}{B}\big(I_c^{\max} - I_c^{\min}\big),\ I_c^{\min} + \frac{b}{B}\big(I_c^{\max} - I_c^{\min}\big) \Big), \quad b = 1, \dots, B.$$

Color and Texture Descriptors

Normalized per-channel histogram.

- For a region $R \subset \Omega$ and a fixed channel c, the normalized histogram is

$$h_c(b) = \frac{1}{|R|} \sum_{(x,y)\in R} 1(I_c(x,y) \in \mathcal{I}_b), \quad b = 1, \ldots, B,$$

  where $1(\cdot)$ is the indicator function.

Building feature vectors.

- Concatenate per-channel histograms:

$$\mathrm{h}(R) = \left(h_1^\top, \ldots, h_C^\top\right)^\top \in \mathbb{R}^{CB},$$

  or define joint histograms over pairs/triples of channels.
- When R is restricted to a border strip, the same construction yields *side-specific color descriptors*:

$$\mathrm{h}\big(B_s(R)\big).$$

Color and Texture Descriptors

Texture features from filter banks.

- We model texture by looking at the responses of a *bank of 2D filters* applied to the image.
- Let $\{F_m\}_{m=1}^{M}$ be a set of 2D filters (e.g. derivatives of Gaussians, oriented edge/line detectors, etc.).
- For each filter F_m, the response is the discrete convolution

$$R_m(x, y) = (I * F_m)(x, y) = \sum_{(u,v)\in\mathbb{Z}^2} I(u, v)\, F_m(x - u, y - v).$$

- If I is multi-channel:
  - either convert to grayscale first,
  - or apply filters per channel and aggregate responses.

Color and Texture Descriptors

Examples of 2D filter families.

- *First-order derivatives of Gaussian.*
  - Gaussian:
$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right).$$
  - Derivative filters:
$$F_\sigma^{(x)}(x,y) = \frac{\partial}{\partial x}G_\sigma(x,y), \qquad F_\sigma^{(y)}(x,y) = \frac{\partial}{\partial y}G_\sigma(x,y).$$
  - Use several scales $\sigma \in \{\sigma_1, \ldots, \sigma_S\}$ to get a bank $\{F_{\sigma_s}^{(x)}, F_{\sigma_s}^{(y)}\}_{s=1}^{S}$.
- *Second-order derivatives / Laplacian-of-Gaussian (LoG).*
  - LoG filter:
$$F_\sigma^{(\mathrm{LoG})}(x,y) = \frac{\partial^2}{\partial x^2}G_\sigma(x,y) + \frac{\partial^2}{\partial y^2}G_\sigma(x,y),$$
    emphasizes blob-like structures.
  - Multiple σ values $\Rightarrow$ multi-scale blob detectors.

Color and Texture Descriptors

- *Oriented edge detectors (e.g. Sobel-like filters).*
  - Standard 3×3 Sobel kernels:

$$F_{\text{Sobel}}^{(x)} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \qquad F_{\text{Sobel}}^{(y)} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}.$$

  - Respond to vertical and horizontal edges, respectively.
  - Additional orientations can be obtained by rotating these kernels or combining them linearly.

Color and Texture Descriptors

Summarizing filter responses over a region.

- Consider a region R (whole piece or border strip).
- For each filter F_m, we have a response image $R_m(x, y)$.
- We summarize the responses on R via simple statistics:
  - *Mean response:*

$$\mu_m = \frac{1}{|R|} \sum_{(x,y)\in R} R_m(x, y).$$

  - *Energy* (e.g. L^2 energy):

$$E_m = \frac{1}{|R|} \sum_{(x,y)\in R} \left|R_m(x, y)\right|^2.$$

  - *Standard deviation:*

$$\sigma_m = \sqrt{\frac{1}{|R|} \sum_{(x,y)\in R} \left(R_m(x, y) - \mu_m\right)^2}.$$

Color and Texture Descriptors

- The texture descriptor for a region R is formed by concatenating the statistics over all filters:

$$\Phi_{\text{tex}}(R) = \mathsf{f}_{\text{tex}}(R) = \left(\mu_1, \ldots, \mu_M,\ \sigma_1, \ldots, \sigma_M,\ E_1, \ldots, E_M\right)^\top.$$

- Applying this construction both on full tiles R and on border strips $B_s(R)$ yields piece-level and side-level texture descriptors, respectively.

Color and Texture Descriptors

Gabor filter-based texture.

- Gabor filters are *localized, oriented, band-pass* filters, often used to model texture in terms of spatial frequency and orientation.
- A 2D Gabor kernel with wavelength λ, orientation θ, phase ψ, standard deviation σ and aspect ratio γ is given by

$$g(x, y) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) \cos\left(2\pi \frac{x'}{\lambda} + \psi\right),$$

  where

$$x' = x\cos\theta + y\sin\theta, \qquad y' = -x\sin\theta + y\cos\theta.$$

- By building a bank of Gabor filters with different orientations θ and wavelengths λ, we obtain a set of responses $\{R_m\}$ on a region R.
- Summarizing these responses over R (e.g. via energies E_m or similar statistics) yields a texture descriptor that is explicitly sensitive to both orientation and spatial frequency.

Local Interest Point Descriptors

- Local interest point descriptors capture the distribution of *gradient-based* features within a region.
- We outline a simplified SIFT-style construction for a generic region Ω; later this will be restricted to border strips $B_s(R_k)$.
- For a pixel $(x, y) \in \Omega$, define the image gradient

$$\nabla I(x, y) = \big(G_x(x, y), G_y(x, y)\big),$$

with magnitude and orientation

$$m(x, y) = \sqrt{G_x(x, y)^2 + G_y(x, y)^2},\ \theta(x, y) = \operatorname{atan2}\big(G_y(x, y), G_x(x, y)\big)$$

Local Interest Point Descriptors

Corner detection on border strips (Harris detector).

- Before computing local descriptors on a side, we must select a set of interest points (corners) within the corresponding border strip.
- Let R_k be a tile with border strip $B_s(R_k)$ for side $s \in \mathcal{S}$.
- For each pixel $(x, y) \in B_s(R_k)$, form the second-moment (structure) matrix over a small window $W_{(x,y)} \subset B_s(R_k)$:

$$M(x, y) = \sum_{(u,v) \in W_{(x,y)}} w(u, v) \begin{bmatrix} G_x(u, v)^2 & G_x(u, v)G_y(u, v) \\ G_x(u, v)G_y(u, v) & G_y(u, v)^2 \end{bmatrix},$$

where $w(u, v)$ is typically a Gaussian weight.

Local Interest Point Descriptors

- Let $\lambda_1(x, y)$ and $\lambda_2(x, y)$ be the eigenvalues of $M(x, y)$, or equivalently define the Harris response

$$R(x, y) = \det M(x, y) - k \left(\operatorname{trace} M(x, y) \right)^2,$$

  with a small constant $k \in [0.04, 0.06]$.
- Intuitively:
  - $R(x, y)$ is large and positive when the gradient changes significantly in *both* directions;
  - such locations correspond to corner-like structures.

- We now use $R(x, y)$ to extract a set of keypoints $\mathcal{K}_s(R_k)$ on each side s of tile R_k.

Local Interest Point Descriptors

Step 1: Response thresholding (candidate corners).

- Choose a threshold $\tau_H > 0$ for the Harris response.
- Consider only pixels in the border strip $B_s(R_k)$ and form the candidate set
$$\mathcal{C}_s(R_k) = \big\{(x, y) \in B_s(R_k) : R(x, y) \geq \tau_H\big\}.$$
- In practice, τ_H can be:
  - an absolute value, or
  - set adaptively (e.g. a fixed percentile of the responses on $B_s(R_k)$)

  so that a reasonable number of candidates is retained on each side.

Local Interest Point Descriptors

Step 2: Non-maximum suppression (corner localization).

- For each candidate $(x, y) \in \mathcal{C}_s(R_k)$, compare $R(x, y)$ to its neighbors in a small window $W_{\text{NMS}}(x, y)$ (e.g. 3×3 or 5×5) restricted to $B_s(R_k)$.
- Keep (x, y) as a keypoint only if

$$R(x, y) = \max_{(u,v) \in W_{\text{NMS}}(x,y)} R(u, v),$$

  i.e. it is a local maximum of the Harris response on that side.
- All other candidates in $\mathcal{C}_s(R_k)$ that are not strict local maxima are discarded.

Local Interest Point Descriptors

Step 3: Refinement and keypoint selection.

- From the remaining local maxima, keep only those points for which a full $K \times K$ descriptor patch lies entirely inside the border strip $B_s(R_k)$.
- Concretely, discard any candidate whose distance to:
  - the tile corners, or
  - the interior boundary of the strip

  is smaller than $\lfloor K/2 \rfloor$ pixels.

- The remaining points form the side-specific keypoint set

$$\mathcal{K}_s(R_k) = \{p_j = (x_j, y_j)\}.$$

- Each $p_j \in \mathcal{K}_s(R_k)$ will serve as the center of a local patch for which a gradient-histogram descriptor is computed.

Local Interest Point Descriptors

Patch-based gradient histogram descriptor: overview

- Let $\mathcal{K}(\Omega)$ denote a set of keypoints detected within a region Ω (e.g. via a corner detector).
- For each keypoint
$$p_j = (x_j, y_j) \in \mathcal{K}(\Omega),$$
we build a local descriptor based on gradient orientations in a square patch around p_j.
- This construction is SIFT-style: it summarizes how gradient energy is distributed across a fixed set of orientation bins.

Local Interest Point Descriptors

Step (i): Local patch extraction

- For each keypoint $p_j = (x_j, y_j)$, extract a square image patch N_j of fixed size $K \times K$ pixels, centered at (x_j, y_j) (for example, $K = 16$ or $K = 24$).
- Let q index pixel locations inside N_j.

Step (ii): Gradient computation

- For each pixel $q \in N_j$, compute discrete image derivatives $G_x(q)$ and $G_y(q)$ (e.g. using finite differences).
- Define the gradient magnitude $m(q)$ and orientation $\theta(q)$ as before.
- Optionally weight each pixel by a Gaussian window centered at p_j:

$$w(q) = \exp\left(-\frac{\|q - p_j\|_2^2}{2\sigma_w^2}\right),$$

where σ_w is proportional to K.

Local Interest Point Descriptors

Step (iii): Orientation histogram construction

- Fix a number of orientation bins B_θ (e.g. $B_\theta = 8$) that partition the interval $[0, 2\pi)$ into equal-width bins:

$$\mathcal{B}_\ell = \Big[2\pi\frac{\ell-1}{B_\theta},\ 2\pi\frac{\ell}{B_\theta}\Big), \qquad \ell = 1, \ldots, B_\theta.$$

- For each pixel $q \in N_j$:
  - find the unique bin index $b(q) \in \{1, \ldots, B_\theta\}$ such that $\theta(q) \in \mathcal{B}_{b(q)}$;
  - add the weighted magnitude $w(q)\, m(q)$ to bin $b(q)$.
- The histogram values are then

$$h_\ell = \sum_{q \in N_j} w(q)\, m(q)\, 1\big(b(q) = \ell\big), \qquad \ell = 1, \ldots, B_\theta,$$

  where $1(\cdot)$ is the indicator function.
- Collecting all bins yields the raw descriptor vector

$$\mathrm{v}_j = (h_1, \ldots, h_{B_\theta})^\top \in \mathbb{R}^{B_\theta}.$$

Local Interest Point Descriptors

Step (iv): Normalization

- To reduce the effect of contrast and illumination changes, normalize the raw histogram:

$$\mathsf{f}_j = \frac{\mathsf{v}_j}{\|\mathsf{v}_j\|_2 + \varepsilon},$$

  where $\varepsilon > 0$ is a small constant.

- The resulting vector

$$\mathsf{f}_j \in \mathbb{R}^{B_\theta}$$

  is the local descriptor at keypoint p_j.

- Repeating this for all $p_j \in \mathcal{K}(\Omega)$ gives a set of local descriptors for region Ω.

Local Interest Point Descriptors

Descriptor set per region and aggregation

- For a region Ω, the construction above yields a variable-length set of descriptors:
$$\mathcal{F}(\Omega) = \{\mathrm{f}_j : p_j \in \mathcal{K}(\Omega)\}.$$
- To obtain a fixed-dimensional descriptor for Ω, we must aggregate $\mathcal{F}(\Omega)$ into a single vector.
- Assuming that $\mathcal{F}(\Omega) = \{\mathrm{f}_1, \ldots, \mathrm{f}_M\}$, the aggregated vector may be defined as:
$$\bar{\mathrm{f}}(\Omega) = \frac{1}{M} \sum_{j=1}^{M} \mathrm{f}_j \in \mathbb{R}^{B_\theta}.$$

Deep CNN Descriptors

Motivation

- In addition to hand-crafted descriptors, we use deep feature representations from a pre-trained convolutional neural network (CNN).
- This allows us to:
  - compare classical features vs. modern deep features,
  - explore combinations of the two for improved performance.
- We treat the CNN as a fixed feature extractor: no training or fine-tuning is required for this project.

Deep CNN Descriptors

CNN model and notation

- Let Φ denote a fixed, pre-trained CNN (e.g. an architecture trained on ImageNet).
- Given an input image
$$X \in \mathbb{R}^{H_0 \times W_0 \times C_0},$$
the network computes a sequence of feature maps (layer outputs)
$$F^{(0)}(X), F^{(1)}(X), \ldots, F^{(L)}(X),$$
where $F^{(0)}(X) = X$.
- For each layer $\ell = 1, \ldots, L$,
$$F^{(\ell)}(X) \in \mathbb{R}^{H_\ell \times W_\ell \times C_\ell}$$
is a 3D tensor: height $\times$ width $\times$ channels.
- You select one or more layers
$$\ell \in \mathcal{L} \subseteq \{1, \ldots, L\}$$

Deep CNN Descriptors

Preprocessing and resizing

- Each CNN expects inputs of a specific size (e.g. $H_0 = W_0 = 224$) and requires specific normalization.
- Let R denote a region (a tile or a border strip).
- Define an operator

$$\mathcal{P} : (\text{image patch}) \to \mathbb{R}^{H_0 \times W_0 \times C_0}$$

that:

  - crops the pixels corresponding to R,
  - resizes the crop to (H_0, W_0) (e.g. bilinear interpolation),
  - applies the required channel-wise normalization (mean/std, scaling, etc.).

- The normalized, resized input for region R is then

$$X_R = \mathcal{P}(R).$$

Deep CNN Descriptors

Feature extraction and spatial pooling

- For a chosen layer $\ell \in \mathcal{L}$, the feature map corresponding to region R is

$$F_R^{(\ell)} = F^{(\ell)}(X_R) \in \mathbb{R}^{H_\ell \times W_\ell \times C_\ell}.$$

- To obtain a fixed-length feature vector from this spatial map, we apply pooling over the spatial dimensions (H_ℓ, W_ℓ).

Global average pooling

- For channel index $c = 1, \ldots, C_\ell$:

$$f_c^{(\ell)}(R) = \frac{1}{H_\ell W_\ell} \sum_{i=1}^{H_\ell} \sum_{j=1}^{W_\ell} F_R^{(\ell)}(i, j, c).$$

- Collecting all channels:

$$\mathrm{f}^{(\ell)}(R) = \left(f_1^{(\ell)}(R), \ldots, f_{C_\ell}^{(\ell)}(R)\right)^\top \in \mathbb{R}^{C_\ell}.$$

Deep CNN Descriptors

Multi-layer deep descriptor

- Often it is useful to combine information from several layers (e.g. low-level textures + higher-level semantics).
- If you select layers

$$\mathcal{L} = \{\ell_1, \ldots, \ell_{|\mathcal{L}|}\},$$

you can concatenate their pooled features:

$$\Phi_{\text{deep}}(R) = \mathsf{f}_{\text{deep}}(R) = \left(\mathsf{f}^{(\ell_1)}(R)^\top, \ \ldots, \ \mathsf{f}^{(\ell_{|\mathcal{L}|})}(R)^\top\right)^\top.$$

- This yields a single multi-scale deep descriptor for region R.

From Piece-Level to Side-Level Descriptors

Idea

- So far, descriptors are defined on generic regions $\Omega \subset \mathbb{Z}^2$ (tiles, patches, border strips).
- We now derive *side-level* descriptors for each tile side $s \in \mathcal{S}$ by restricting region-based descriptors to the border strips $B_s(R_k)$.
- This gives one descriptor (or descriptor vector) per side, which is crucial for adjacency modeling between tiles.

From Piece-Level to Side-Level Descriptors

General construction

- Let $\mathcal{D}$ be the set of descriptor families used at the piece level.
- For each $d \in \mathcal{D}$, let

$$\Phi_d : (\text{image region}) \to \mathbb{R}^{D_d}, \qquad R \mapsto \Phi_d(R)$$

  be the corresponding descriptor mapping.
- To obtain a side-level descriptor for side s of tile R_k, simply restrict the region to the border strip $B_s(R_k)$:

$$\mathrm{f}_{d,s}(R_k) = \Phi_d\big(B_s(R_k)\big) \in \mathbb{R}^{D_d}.$$

- The definition of Φ_d does not change; only its spatial support is restricted to a thin band near the tile side.

From Piece-Level to Side-Level Descriptors

Combining multiple side-level families

- For each side (R_k, s) you may concatenate multiple descriptor families into a single vector:

$$\mathsf{f}_s(R_k) = \big(\alpha_{\mathrm{col}}\,\mathsf{f}_{\mathrm{col},s}(R_k)^\top,\ \alpha_{\mathrm{tex}}\,\mathsf{f}_{\mathrm{tex},s}(R_k)^\top,$$

$$\alpha_{\mathrm{loc}}\,\mathsf{f}_{\mathrm{loc},s}(R_k)^\top,\ \alpha_{\mathrm{deep}}\,\mathsf{f}_{\mathrm{deep},s}(R_k)^\top\big)^\top.$$

- Non-negative weights $\alpha_{\mathrm{col}}, \alpha_{\mathrm{tex}}, \alpha_{\mathrm{loc}}, \alpha_{\mathrm{deep}}$ control the relative contribution of each descriptor family.
- This creates a unified side-level representation that can be plugged directly into adjacency scoring models.

From Piece-Level to Side-Level Descriptors

Side-level color descriptors

- Let Φ_{col} be the color histogram mapping which maps a region R to its normalized (possibly multi-channel) color histogram.
- The side-level color descriptor for side s of tile R_k is obtained by restricting to $B_s(R_k)$:

$$\mathrm{f}_{\text{col},s}(R_k) = \Phi_{\text{col}}\big(B_s(R_k)\big).$$

- These descriptors capture how color statistics behave right at the potential interface between neighboring pieces, and provide a strong baseline for adjacency estimation.

From Piece-Level to Side-Level Descriptors

Side-level texture and deep descriptors

Side-level texture descriptors

- Let Φ_{tex} be the texture descriptor mapping built from filter-bank responses.
- The side-level texture descriptor for side s of tile R_k is

$$\mathrm{f}_{\text{tex},s}(R_k) = \Phi_{\text{tex}}\big(B_s(R_k)\big).$$

- Here, filter responses are computed only for $(x, y) \in B_s(R_k)$.

Side-level deep CNN descriptors

- Let Φ_{deep} be the deep CNN descriptor mapping.
- The side-level deep descriptor is obtained as

$$\mathrm{f}_{\text{deep},s}(R_k) = \Phi_{\text{deep}}\big(B_s(R_k)\big),$$

a high-level deep feature vector focused on the visual content near side s.

From Piece-Level to Side-Level Descriptors

Local interest point descriptors on borders

- Detect keypoints within $B_s(R_k)$ and denote their set by

$$\mathcal{K}_s(R_k) \subset B_s(R_k).$$

- For each keypoint $p_j = (x_j, y_j) \in \mathcal{K}_s(R_k)$:
  - compute a SIFT-style descriptor f_j;
  - compute a normalized shift coordinate τ_j that encodes its position along side s:

$$\tau_j = \begin{cases} \dfrac{y_j - y_{\min}}{y_{\max} - y_{\min}}, & \text{top/bottom sides,} \\ \dfrac{x_j - x_{\min}}{x_{\max} - x_{\min}}, & \text{left/right sides,} \end{cases}$$

  so that $\tau_j \in [0, 1]$.

From Piece-Level to Side-Level Descriptors

Aggregating local border descriptors

- For each side s of tile R_k, we obtain a variable-length set

$$\mathcal{F}_s(R_k) = \left\{(\mathrm{f}_j, \tau_j) : p_j \in \mathcal{K}_s(R_k)\right\}.$$

- To get a fixed-dimensional side-specific descriptor $\mathrm{f}_{\mathrm{loc},s}(R_k)$, we aggregate the f_j in $\mathcal{F}_s(R_k)$ as:

$$\mathrm{f}_{\mathrm{loc},s}(R_k) = \frac{1}{M}\sum_{j=1}^{M} \mathrm{f}_j.$$

From Piece-Level to Side-Level Descriptors

Summary

- For each tile R_k and side $s \in \mathcal{S}$, you may construct the collection

$$\left\{ \mathrm{f}_{\mathrm{col},s}(R_k),\ \mathrm{f}_{\mathrm{tex},s}(R_k),\ \mathrm{f}_{\mathrm{loc},s}(R_k),\ \mathrm{f}_{\mathrm{deep},s}(R_k) \right\}$$

  by applying your strongest region-based descriptors to the border strip $B_s(R_k)$.
- These side-level descriptors provide the main inputs for adjacency modeling in the global reconstruction stage.

Adjacency Modeling

Goal

- We have:
  - side-level feature vectors for each piece side,
  - tile-level (piece-level) descriptors for each piece.
- We want to define *compatibility scores* between candidate neighboring sides:

$$C\big((i,s),(j,t)\big) \in \mathbb{R},$$

  measuring how likely it is that side s of piece i was adjacent to side t of piece j in the original image.
- Both local border evidence and global tile similarity should contribute to this score.

Adjacency Modeling

Notation

- Pieces are indexed by $i, j \in \{1, \ldots, N\}$.
- Sides are indexed by

$$s, t \in \mathcal{S} = \{\text{top}, \text{bottom}, \text{left}, \text{right}\}.$$

- For each descriptor family d (color, texture, local, deep, etc.) and side s of piece i, we have a side-level feature vector

$$\mathrm{f}_{d,s}(i) \in \mathbb{R}^{D_d},$$

obtained from the border strip $B_s(R_i)$.

- For the same family d, we also have a tile-level descriptor on the whole region R_i:

$$\mathrm{f}_d^{\text{tile}}(i) \in \mathbb{R}^{\tilde{D}_d},$$

using the same mapping Φ_d but on R_i (global color, global texture, deep features, etc.).

Adjacency Modeling

Side-level and tile-level distances

For each descriptor family d:

- *Side-level distance* between side s of piece i and side t of piece j:

$$d_d^{\text{side}}\big((i,s),(j,t)\big) = d_d\big(\mathrm{f}_{d,s}(i),\, \mathrm{f}_{d,t}(j)\big) \;\geq 0.$$

- *Tile-level distance* between the whole pieces i and j:

$$d_d^{\text{tile}}(i,j) = d_d\big(\mathrm{f}_d^{\text{tile}}(i),\, \mathrm{f}_d^{\text{tile}}(j)\big) \;\geq 0.$$

- Here $d_d(\cdot,\cdot)$ is any suitable distance for family d.

Adjacency Modeling

From distances to similarities

- Distances are converted to similarities via a decreasing function g_d:

$$C_d^{\text{side}}\big((i,s),(j,t)\big) = g_d\Big(d_d^{\text{side}}\big((i,s),(j,t)\big)\Big),$$
$$C_d^{\text{tile}}(i,j) = g_d\big(d_d^{\text{tile}}(i,j)\big).$$

Typical choices:

- *Gaussian-like similarity*:

$$g_d(d) = \exp(-\lambda_d d^2), \qquad \lambda_d > 0.$$

- *Cosine similarity* (especially for deep features), e.g. side-level:

$$C_d^{\text{side}}\big((i,s),(j,t)\big) = \frac{\mathrm{f}_{d,s}(i)^\top \mathrm{f}_{d,t}(j)}{\|\mathrm{f}_{d,s}(i)\|_2 \, \|\mathrm{f}_{d,t}(j)\|_2},$$

and similarly for $C_d^{\text{tile}}(i,j)$ using $\mathrm{f}_d^{\text{tile}}(i)$ and $\mathrm{f}_d^{\text{tile}}(j)$.

Adjacency Modeling

Descriptor-specific compatibility terms

- For each descriptor family d, combine side-level and tile-level similarities into a single term:

$$C_d\big((i,s),(j,t)\big) = w_d^{\text{side}}\, C_d^{\text{side}}\big((i,s),(j,t)\big) + w_d^{\text{tile}}\, C_d^{\text{tile}}(i,j),$$

- where $w_d^{\text{side}}, w_d^{\text{tile}} \geq 0$ control the relative influence of:
  - local border evidence (side-level descriptors),
  - global tile similarity (tile-level descriptors).

Adjacency Modeling

Combined compatibility score

- The overall compatibility between side s of piece i and side t of piece j sums across descriptor families:

$$C\big((i,s),(j,t)\big) = \sum_{d\in\mathcal{D}} \alpha_d \, C_d\big((i,s),(j,t)\big),$$

  where $\alpha_d \geq 0$ are global weights controlling the importance of each family (color, texture, deep, local, ...).

- In expanded form:

$$C\big((i,s),(j,t)\big) = \sum_{d\in\mathcal{D}} \alpha_d \Big(w_d^{\text{side}} \, C_d^{\text{side}}\big((i,s),(j,t)\big) + w_d^{\text{tile}} \, C_d^{\text{tile}}(i,j) \Big).$$

- Note: $C((i,s),(j,t))$ is generally *directed*: right side of i vs left side of j is not the same as left side of i vs right side of j, especially under rotations.

Adjacency Modeling

Learning or tuning the weights

- Weights α_d, w_d^{side}, w_d^{tile} can be:
  - *manually tuned* (e.g. higher weight on color and deep features, moderate on texture, etc.);
  - or *learned* from data via a simple supervised model.

Learning procedure (sketch)

1. Use the ground-truth puzzle to construct:
   - *positive* examples: true neighboring side pairs $((i, s), (j, t))$,
   - *negative* examples: randomly sampled non-neighboring side pairs.
2. For each pair, compute a feature vector collecting all descriptor-specific similarities, e.g.

$$\mathrm{z}((i, s), (j, t)) = \left(C_{\text{color}}^{\text{side}}, C_{\text{color}}^{\text{tile}}, C_{\text{texture}}^{\text{side}}, C_{\text{texture}}^{\text{tile}}, C_{\text{deep}}^{\text{side}}, C_{\text{deep}}^{\text{tile}}, \dots\right)^{\top}.$$

3. Fit a linear classifier (e.g. logistic regression, linear SVM):

$$S((i, s), (j, t)) = \mathrm{w}^{\top}\mathrm{z}((i, s), (j, t)) + b.$$

Global Reconstruction Algorithm

Goal

- Use:
  - side-level compatibility scores,
  - global piece-level descriptors,

  to reconstruct the puzzle.
- Formulate reconstruction as a discrete optimization problem over:
  - *placements* of pieces on the grid,
  - *orientations* of all pieces.
- Search for the configuration that maximizes a global objective combining adjacency and placement cues.

Global Reconstruction Algorithm

Configuration space

- Let $\mathcal{G}$ be the set of grid positions and $N = |\mathcal{G}|$ the number of pieces.
- Let $\mathcal{A}$ be the finite set of allowed orientations, e.g.

$$\mathcal{A} = \{0^\circ, 90^\circ, 180^\circ, 270^\circ\}.$$

- A configuration consists of:
  - a permutation
  $$\pi : \{1, \ldots, N\} \to \mathcal{G},$$
  placing each piece at a unique grid position;
  - an orientation
  $$\theta_k \in \mathcal{A}, \qquad k = 1, \ldots, N,$$
  for each piece.
- The space of all $(\pi, \{\theta_k\})$ is huge $\Rightarrow$ approximate optimization is needed.

Global Reconstruction Algorithm

Goal

- Reconstruct the original image by deciding
  - where each piece should be placed in the grid, and
  - which orientation each piece should have,
- using the side-level compatibility scores.

Configuration space

- $\mathcal{G}$: set of grid positions, $N = |\mathcal{G}|$ pieces.
- $\mathcal{A}$: set of allowed orientations (e.g. $\mathcal{A} = \{0°, 90°, 180°, 270°\}$).
- A configuration consists of:
  - a permutation $\pi : \{1, \ldots, N\} \to \mathcal{G}$,
  - an orientation $\theta_k \in \mathcal{A}$ for each piece k.

The configuration space $\{(\pi, \{\theta_k\})\}$ is huge, so approximate optimization methods are needed.

Side-based Adjacency Score

Neighboring grid positions

- Use 4-neighborhood connectivity (up, down, left, right).
- Let $\mathcal{E}$ be the set of all neighboring grid pairs:

$$\mathcal{E} = \left\{ \big((r, c), (r', c')\big) \in \mathcal{G} \times \mathcal{G} : |r - r'| + |c - c'| = 1 \right\}.$$

- Each element of $\mathcal{E}$ represents two adjacent cells.

Which sides touch? For a configuration $(\pi, \{\theta_k\})$ and a pair $\big((r, c), (r', c')\big) \in \mathcal{E}$:

- i is the piece with $\pi(i) = (r, c)$,
- j is the piece with $\pi(j) = (r', c')$,
- the relative positions of (r, c) and (r', c') and the orientations θ_i, θ_j determine which side s of piece i faces which side t of piece j (e.g. right side of i vs left side of j).

Total Adjacency Score

Side compatibility

- For each touching side pair (i, s) and (j, t) we compute the compatibility score:

$$C\big((i, s), (j, t)\big)$$

  defined earlier.

Adjacency-based score of a configuration

$$F_{\text{adj}}(\pi, \{\theta_k\}) = \sum_{\big((r,c),(r',c')\big)\in\mathcal{E}} C\big((i, s), (j, t)\big),$$

where, for each neighbor pair $\big((r, c), (r', c')\big)$, the indices i, j, s, t are determined by π and $\{\theta_k\}$ as described in the previous slide.

Intuition

- F_{adj} is large when many neighboring pairs in the grid correspond to visually compatible sides.

Adjacency-based Objective Function

Objective

- In our basic formulation, the total quality of a configuration is exactly its adjacency score:

$$F_{\text{tot}}(\pi, \{\theta_k\}) = F_{\text{adj}}(\pi, \{\theta_k\}).$$

Global reconstruction problem

$$\max_{\pi, \{\theta_k\}} \; F_{\text{tot}}(\pi, \{\theta_k\})$$

subject to

$$\pi \text{ is a permutation of } \{1, \ldots, N\}, \qquad \theta_k \in \mathcal{A} \text{ for all } k.$$

Constraints

- One piece per grid cell.
- One orientation per piece.

Integer Linear Programming (ILP) Formulations

Binary decision variables

- For small puzzles, introduce

$$x_{k,g,a} = \begin{cases} 1, & \text{if piece } k \text{ is at grid position } g \text{ with orientation } a, \\ 0, & \text{otherwise,} \end{cases}$$

for all $k \in \{1, \ldots, N\}$, $g \in \mathcal{G}$, $a \in \mathcal{A}$.

Assignment constraints

$$\sum_{g \in \mathcal{G}} \sum_{a \in \mathcal{A}} x_{k,g,a} = 1 \quad \forall k,$$

$$\sum_{k=1}^{N} \sum_{a \in \mathcal{A}} x_{k,g,a} = 1 \quad \forall g \in \mathcal{G}.$$

Precompute adjacency scores $C_{k,\ell}^{g,g',a,b}$ for neighbor pairs $(g, g') \in \mathcal{E}$, pieces (k, ℓ) and orientations (a, b).

Integer Linear Programming (ILP) Formulations

Quadratic objective

$$\max_{x} \sum_{(g,g')\in\mathcal{E}} \sum_{k,\ell=1}^{N} \sum_{a,b\in\mathcal{A}} C_{k,\ell}^{g,g',a,b}\, x_{k,g,a}\, x_{\ell,g',b},$$

subject to the assignment constraints.

Remarks

- This is a quadratic assignment / integer quadratic program.
- Exact solution is typically feasible only for small puzzles.
- Useful mainly as a mathematical reference.

Heuristic Search and Practical Expectations

Heuristic search

- For realistic puzzle sizes, use approximate search over $(\pi, \{\theta_k\})$, e.g. local search:
  - swap the positions of two pieces (with or without rotating them),
  - rotate a single piece in place,
  - move a piece to a different grid position.
- Accept moves that increase F_{tot} and stop when no simple move improves the objective.